Downloaded from orbit.dtu.dk on: May 05, 2023

DTU Library

High Performance Network Evaluation and Testing

Pilimon, Artur

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Pilimon, A. (2018). *High Performance Network Evaluation and Testing*. DTU - Department of Photonics Engineering.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

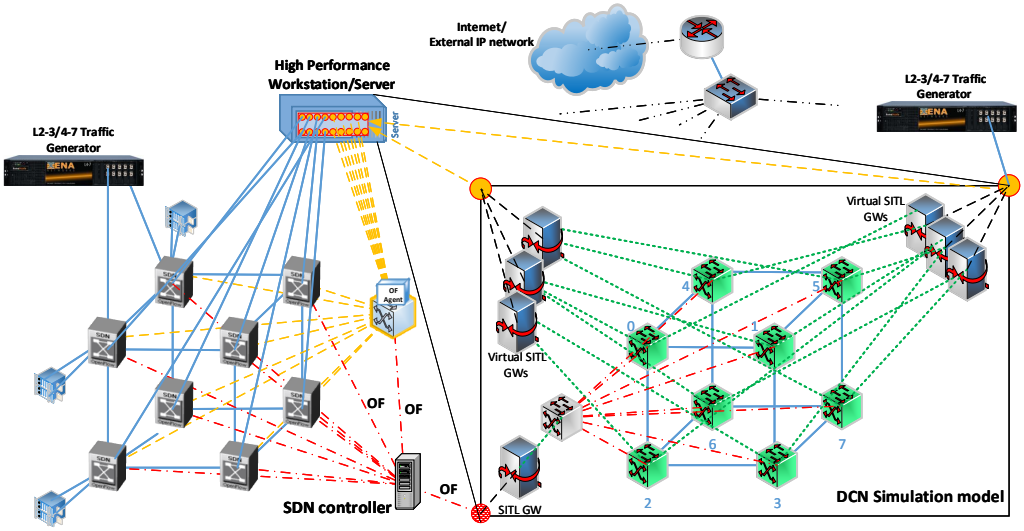
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

PhD Thesis

High Performance Network Evaluation and Testing

Artur Pilimon

Supervisor: Assoc. Prof. Sarah Renée Ruepp
Co-supervisor: Assoc. Prof. Michael Stübert Berger



Technical University of Denmark
Kgs. Lyngby, Denmark 2018

To my dear parents, Andrej and Larisa, for their priceless support and wisdom. To my sister Diana, for shaping my personality in the way I am. To my darling, Linda, for her endless love, patience and encouragement.

Technical University of Denmark
DTU Fotonik, Department of Photonics Engineering
Networks Technology and Service Platforms
Ørsteds Plads 343
2800 Kgs. Lyngby
DENMARK
Tel: (+45) 4525 6352
Web: www.fotonik.dtu.dk/

Contents

Abstract

This thesis focuses on the testing and performance evaluation aspects of High Speed Networks, covering such areas as Layer 4-7 testing, focusing on protocol-level performance evaluation of long-distance high-speed communication, as well as performance characterization and testing of Data Center Networks (DCNs). From a broad range of open research questions in these areas, a subset of them is investigated in this work, such as: Layer 4 Transmission Control Protocol (TCP) Congestion Control (CC) challenges and algorithmic properties of a high-speed TCP CUBIC extension with additional supporting algorithms, and addressing a scalability aspect of performance evaluation in DCNs from an experimental perspective (defining a methodology for DCN testing at scale). Furthermore, research activities, presented in this work, include testing and analysis of Software Defined Networking (SDN) related performance aspects, namely flow-rule placement in SDN switches and SDN Traffic Engineering (TE) capabilities for DCNs, as well as energy efficiency aspects in DCNs with optical switching.

A dominant portion of global data traffic is transported by the TCP protocol. Therefore, a functional and stable TCP connection engine is of paramount importance to ensure reliable and in-order end-to-end data delivery for a diverse range of applications and services. However, the core functionality of the TCP protocol cannot efficiency handle this task alone, and a large set of additional extensions and algorithms are used in combination to enable scalable and adaptive data transmission. In this context, Congestion Control extensions of TCP, such as high-speed CUBIC (used in Linux and Windows 10 Operating Systems), are playing a fundamental role in providing a means of adaptive data transmission in network environments, potentially shared among a large number of different types of flows, as well as preventing such operational network disruptions as congestion collapse. The problem that arises in this situation is that there is still no uniform opinion or agreement on which specific algorithms and extensions enabled with TCP produce the best results in terms of communication stability, adaptability to diverse operational conditions and network environments (e.g., wireless, wired, Internet, DCNs, etc.), resource utilization efficiency, as well as fair bandwidth sharing. As a result, there are many open questions, requiring comprehensive research. Therefore, this work contributes with additional analysis of such aspects as: CC in modern high-speed networks, such as the Internet, as well as algorithmic robustness and packet loss recovery

efficiency of high speed TCP CUBIC connections.

Data Centers (DCs) have truly become the backbone of the global economy, providing a mission-critical infrastructure for a broad range of different applications and services with very diverse Quality of Service (QoS) requirements. Tremendous growth of the global DC IP traffic facilitated active research of new architectural and technological solutions in order to optimize the operational efficiency of the existing infrastructures as well as to build more scalable, energy- and resource-efficient, future-proof DCN architectures and protocols. The question that arises is how do we test and verify that new solutions and innovative research ideas are, indeed, efficient, applicable at large scale and can be transferred to real production DCN environments, while not having access to such facilities for experimental and testing purposes? This is the question that this thesis attempts to answer by proposing a novel hybrid physical-simulated electrical-optical and SDN-controlled DCN testbed for performance and scalability studies and active experimentation.

Another important topic, investigated in this thesis, is SDN-related testing and performance evaluation. The growth of scale and complexity of DCN architectures facilitated a need for new optimized ways of resource (compute, storage, networking) management, higher degree of control and programmability of the data plane (network gear) in a more vendor-agnostic, centralized manner. SDN and Cloud Computing paradigms were introduced to address many of these challenges. However, this relatively new technological shift also introduced new challenges in terms of the scalability of the centralized control plane, security concerns, functional capabilities and limitations of the open control interfaces (e.g., OpenFlow), etc. This work focuses on a set of specific SDN-related performance evaluation aspects, such as: performance characterization of a novel flow-rule placement algorithm for SDN switches, focusing on the optimization of resource usage in hybrid (hardware and software) Flow tables, as well as an analysis of the TE requirements for DCNs and the capabilities provided by SDN in this context.

Finally yet importantly, Energy Efficiency aspects in DCNs are explored in this work by evaluating the impact of optical circuit switching, selectively applied on different network layers of a set of considered DCN topologies, such as a traditional three-tier Tree, a Fat-Tree and a Ring-based structure, enhanced by the Wavelength Division Multiplexing (WDM) capabilities. A simulation-based DCN dimensioning is performed and these results are used as an input to a defined power consumption model.

Resumé

Denne afhandling fokuserer på test- og ydelsesaspekterne af højhastighedsnetværk, og dækker områder som Layer 4-7 test, med fokus på evaluering af højhastighedskommunikation på langdistanceområdet samt ydelseskarakterisering og test af Data Center Networks (DCN'er). Fra en bred vifte af åbne forskningsspørgsmål på disse områder undersøges en delmængde af disse i dette arbejde, såsom: Layer 4 Transmission Control Protocol (TCP), Congestion Control (CC), udfordringer og algoritmiske egenskaber ved en højhastigheds TCP CUBIC udvidelse med yderligere understøttende algoritmer. Derudover adresseres et skalerbarhedsaspekt ved ydelsesevaluering i DCN'er fra et eksperimentelt perspektiv (definere en metode til DCN-test i skala). Desuden omfatter forskningsaktiviteter, der præsenteres i dette arbejde, test og analyse af SDN-relaterede ydelsesaspekter, nemlig flowregulering i SDN-switches og SDN Traffic Engineering (TE) -funktioner til DCN'er samt energieffektivitetsaspekter i DCN'er med optisk switching.

En dominerende del af den globale datatrafik transporteres af TCP-protokollen. Derfor er en funktionel og stabil TCP-forbindelse af afgørende betydning for at sikre pålidelig og korrekt end-to-end dataudlevering til en bred vifte af applikationer og tjenester. Kernefunktionaliteten i TCP-protokollen kan imidlertid ikke levere denne opgave alene, og et stort sæt yderligere udvidelser og algoritmer anvendes i kombination for at muliggøre skalerbar og adaptiv dataoverførsel. I denne sammenhæng spiller Congestion Control-udvidelser af TCP, såsom CUBIC (anvendt i Linux og Windows 10-operativsystemer) en afgørende rolle i at tilvejebringe et middel til adaptiv dataoverførsel i netværksmiljøer, som muligvis deles blandt et stort antal af forskellige typer af strømme, samt at forhindre trængsel og sammenbrud. Problemet der opstår i denne situation er, at der stadig ikke er nogen ensartet aftale om, hvilke specifikke algoritmer og udvidelser, der er aktiveret med TCP, giver de bedste resultater med hensyn til kommunikationsstabilitet, tilpasningsevne til forskellige driftsforhold og netværksmiljøer (f.eks. Trådløst, kablet, Internet, DCN'er osv.), Ressourceudnyttelseseffektivitet samt fair båndbreddedeling. Som følge heraf er der mange åbne spørgsmål, der kræver omfattende forskning. Derfor bidrager dette arbejde med yderligere analyse af sådanne aspekter som: CC i moderne højhastighedsnet, som f.eks. Internettet, samt algoritmisk robusthed og pakkefejlgendannelseseffektivitet af højhastigheds-TCP CUBIC-forbindelser.

Data Centers (DCs) er virkelig blevet rygraden i den globale økonomi, hvilket giver

en missionskritisk infrastruktur til en bred vifte af forskellige applikationer og tjenester med meget forskellige Quality of Service (QoS) krav. En enorm vækst i den globale DC IP-trafik muliggjorde aktiv forskning i nye teknologiske løsninger for at optimere de eksisterende infrastrukturers driftseffektivitet samt at opbygge mere skalerbare, energi- og ressourceeffektive fremtidssikrede DCN-arkitekturer og protokoller. Det spørgsmål, der opstår, er, hvordan vi tester og verificerer, at nye løsninger og innovative forskningsideer rent faktisk er effektive, anvendelige i stor skala og kan overføres til DCN-miljøer i ægte produktion, uden at der er adgang til sådanne faciliteter til forsøgs- og testformål? Dette er spørgsmålet, som denne afhandling forsøger at besvare ved at foreslå en ny hybrid fysisk simuleret elektrisk-optisk og SDN-styret DCN testbed for performance- og skalerbarhedsundersøgelser og aktive eksperimenter

Et andet vigtigt emne, undersøgt i denne afhandling, er SDN-relateret test og ydelsesevaluering. Størrelsen og kompleksiteten af DCN-arkitekturer skabte et behov for nye, optimerede måder til ressourceforvaltning (beregning, lagring, netværksstyring), højere grad af styring og programmerbarhed af dataplanet (netværksudstyr) . SDN og Cloud Computing-paradigmer blev introduceret for at løse mange af disse udfordringer. Dette relativt nye teknologiske skift indførte dog også nye udfordringer med hensyn til skalerbarheden af det centraliserede kontrolplan, sikkerhedsproblemer, funktionelle evner og begrænsninger af de åbne kontrolflader (f.eks. OpenFlow) osv. Dette arbejde fokuserer på et sæt specifikke SDN-relaterede ydelsesevalueringsaspekter, såsom: ydelseskarakterisering af en ny flowreguleringsplaceringsalgoritme til SDN-switches, der fokuserer på optimering af ressourceforbrug i hybrid (hardware og software) flow tabeller samt en analyse af TE krav for DCN'er og de muligheder, som SDN tilbyder i denne sammenhæng.

Endelig udforskes energieffektivitetsaspekterne i DCN'er i dette arbejde ved at evaluere virkningen af optisk kredsløbskobling, der selektivt anvendes på forskellige netværkslag af et sæt af typiske DCN topologier, såsom et traditionelt tre-tier-træ, en Fat- Tree og en Ring-baseret struktur, udvidet med Wavelength Division Multiplexing (WDM). En simuleringsbaseret DCN-dimensionering udføres, og disse resultater bruges som input til en defineret model af strømforbruget.

List of Figures

List of Tables

List of Acronyms

ABC	Appropriate Byte Counting
ACK	Acknowledgement
ACKN	Acknowledgement Number
AIMD	Additive Increase Multiplicative Decrease
APIs	Application Programming Interfaces
AQM	Active Queue Management
ARPA	Advanced Research Projects Agency
AWND	Advertised Window
AWS	Amazon Web Services
BBR	Bottleneck Bandwidth and Round-trip propagation time
BBW	Bottleneck Bandwidth
BDP	Bandwidth-Delay Product
BIC	Binary Increase Congestion control
BOR	Bandwidth Oversubscription Ratio
bps	bits per second
BUE	Bandwidth Utilization Efficiency
CA	Congestion Avoidance
CAGR	Compound Annual Growth Rate
CAPEX	CApital EXpenditures
CC	Congestion Control
CCA-RTC	Congestion Control Algorithm for Real-Time Communication
CDN	Content Delivery Network
CORE	Common Open Research Emulator
COSIGN	Combining Optics and SDN In next Generation data centre Networks
CPU	Central Processing Unit
CRB	Conservative Reduction Bound
CSPs	Content Service Providers

CTCP	Compound TCP
CWND	Congestion Window
DACK	Duplicate ACKnowledgment
DACKs	Duplicate ACKnowledgments
DC	Data Center
DCCP	Datagram Congestion Control Protocol
DCN	Data Center Network
DCNI	Data Center Network Interconnect
DCNs	Data Center Networks
DCs	Data Centers
DevOps	Development and Operations
DL	Deep Learning
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DSACK	Duplicate-Selective Acknowledgment
DSL	Digital Subscriber Line
DT	Drop Tail
DUT	Device Under Test
DWDM	Dense Wavelength Division Multiplexing
DWND	Delay Window
EC2	Elastic Compute Cloud
ECN	Explicit Congestion Notification
EE	Energy Efficiency
ER	Early Retransmit
FAST	Fast AQM Scalable TCP
FCT	Flow Completion Time
FIN	Finish
FMC	Fixed-Mobile Convergence
FPGA	Field-Programmable Gate Array
FREC	Fast Recovery
FRTX	Fast Retransmit
FTTH	Fiber To The Home
GCI	Global Cloud Index
GFAST	Generalized FAST
H-TCP	Hamilton-TCP
HPC	High Performance Computing
HPCC	High-Performance Computing Cluster
HSTCP	High Speed TCP
HSTCP-LP	High Speed TCP for Low Priority
HVAC	Heating, Ventilation and Air Conditioting
I/O	Input/Output
IaaS	Infrastructure-as-a-Service

ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoE	Internet of Everything
IoT	Internet of Things
IP	Internet Protocol
IPC	Inter-Process Communication
ISN	Initial Sequence Number
ISPs	Internet Service Providers
IST	Integrated System Test
IT	Information Technology
IW	Initial Window
kB/s	kilobyte per second
LB	Load Balancing
LBNL	Lawrence Berkeley National Laboratory
LFN	Long Fat Network
LP	Latency Performance
LT	Limited Transmit
LTE	Long Term Evolution
MIMD	Multiplicative Increase Multiplicative Decrease
ML	Machine Learning
MSL	Maximum Segment Lifetime
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
MulTCP	Multiple TCP
NETCONF	NETwork CONFiguration
NFV	Network Function Virtualization
NIC	Network Interface Card
NUT	Network Under Test
NV	Network Virtualization
OF	OpenFlow
OF-Config	OpenFlow Management and Configuration Protocol
OOO	Out Of Order
OPEX	OPerational EXpenditures
OS	Operating System
OSI	Open Systems Interconnection
OTN	Optical Transport Network
P2P	Peer-to-Peer
PaaS	Platform-as-a-Service
PAWS	Protection Against Wrapped Sequences
PI	Performance Indicator
PLRE	Packet Loss Recovery Efficiency

PoC	Proof-of-Concept
pps	packets per second
PRR	Proportional Rate Reduction
PUE	Power Usage Effectiveness
QoE	Quality of Experience
QoS	Quality of Service
QUIC	Quick UDP Internet Connection
RACK	Recent Acknowledgement
RAM	Random Access Memory
RCP	Rate Control Protocol
REM	Random Exponential Marking
RTC	Real-Time Communication
RTk	Real-Time kernel
RTO	Retransmission Timeout
RTprop	Round-Trip propagation delay
RTT	Round Trip Time
SaaS	Software-as-a-Service
SACK	Selective ACKnowledgement
SCTP	Stream Control Transmission Protocol
SD-WAN	Software-Defined-Wide Area Network
SDN	Software Defined Networking
SDOs	Standards Development Organizations
SIAD	Scalable Increase Adaptive Decrease
SITL	System-in-the-Loop
SN	Sequence Number
SNMP	Simple Network Management Protocol
SOTA	State-of-the-art
SS	Slow Start
SS7	Signalling System 7
SSRB	Slow Start Reduction Bound
SSTHOLD	Slow Start Threshold
STCP	Scalable TCP
STP	Spanning Tree Protocol
SUT	Service Under Test
SW	Sliding Window
SWND	Send Window
SYN	Synchronize
TCAM	Ternary Content Addressable Memory
TCB	TCP Connection Block
TCP	Transmission Control Protocol
TCP-LP	TCP Low Priority
TE	Traffic Engineering

TFRC	TCP-Friendly Rate Control
TR	Transmission Round
TS	Timestamp
UCB	University of California, Berkeley
UDP	User Datagram Protocol
VCP	Variable-structure Congestion control Protocol
VM	Virtual Machine
VNF	Virtual Network Function
VoD	Video-on-Demand
VPC	Virtual Private Cloud
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WS	Window Scaling
XaaS	Anything-as-a-Service
XCP	Explicit Control Protocol
YeAH	Yet Another High-speed TCP
ZB	Zetta Bytes

Ph.D. Publications

The following scientific papers are the outcome of this Ph.D. project and are included as Part II of this thesis.

Published/presented/accepted:

[Paper A/Poster] **A. Pilimon**, "High Speed Network Evaluation and Testing", in *5th PhD School on Traffic Monitoring and Analysis (TMA)*, Poster Presentation, Barcelona, Spain, 21-22 April, 2015.

[Paper B] **A. Pilimon**, S. Ruepp and M. S. Berger, "Robustness of multiple high speed TCP CUBIC connections under severe operating conditions", in *Proc. IEEE 14th International Symposium on Network Computing and Applications (NCA)*, Boston, MA, USA, 28-30 Sept., 2015, pp. 76-80.

[Paper C] **A. Pilimon**, A. Zeimpeki, A. M. Fagertun and S. Ruepp, "Energy efficiency benefits of introducing optical switching in Data Center Networks", in *Proc. of Workshop on Computing, Networking and Communications (CNC)*, Santa Clara, CA, USA, 26-29 Jan., 2017, pp. 891-895.

[Paper D] S. Ruepp*, **A. Pilimon***, J. Thrane, Michael Galili, Michael Berger and Lars Dittmann, "Combining hardware and simulation for datacenter scaling studies", in *Proc. International Conference on Optical Network Design and Modeling (ONDM)*, Budapest, Hungary, 15-18 May, 2017, pp. 1-6.

[Paper E] **A. Pilimon** and S. Ruepp, "A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks", presented at *IEEE International Conference on Computing, Networking and Communications (ICNC)*, Maui, Hawaii, USA, March 5-8, 2018, pp. 1-5.

[Paper G] A. Mimidis-Kentis*, **A. Pilimon***, J. Soler, M. Berger and S. Ruepp, "A Novel Algorithm for Flow-Rule Placement in SDN Switches", accepted to *4th IEEE Conference on Network Softwarization (NetSoft)*, Montreal, Canada, 25-29 June, 2018, 9 p.

[Paper H] **A. Pilimon**, A. Mimidis-Kentis, S. Ruepp and L. Dittmann, "Analysis of Traffic Engineering capabilities for SDN-based Data Center Networks", presented at *The Second IEEE international workshop on Software Defined Networks and Network Function Virtualization (SDN-NFV)*, Barcelona, Spain, 23-26 April, 2018, 6 p.

[Paper I] O. Sørensen* (Xena Networks ApS), **A. Pilimon*** (DTU) and S. Ruepp (DTU), "Evaluate Data Center Network Performance", *Technical White Paper*, pp. 1-11, Jan. 2018.[Online]. Available: https://xenanetworks.com/evaluate-data-center-network-performance-white-paper/.

Submitted/under review:

[Paper F] **A. Pilimon**, S. Ruepp and L. Dittmann, "A Novel Hybrid Testbed Combining Optics and SDN for Topology-Agnostic Datacenter Network Evaluation", submitted to *Computer Networks Journal*, pp. 1-5, April 2018.

Acknowledgements

This work wouldn't be possible without the support, encouragement and friendship of a number of interesting people, which I had a chance to interact with throughout my journey as a Ph.D. student at the Department of Photonics Engineering.

First of all, I would like to express my gratitude to my main and co-supervisors, namely Sarah Renée Ruepp and Michael Stübert Berger, respectively, for sharing their invaluable experience and providing me with useful suggestions that helped to shape this work in the way it is. In addition, I am thankful to them for giving me a great opportunity to immerse into an exciting and, at the same time, challenging world of network research, both in theoretical and experimental realms, and for charging me with motivation to do exciting work despite some associated ups and downs.

Furthermore, I want to thank all the co-authors of my research papers for their valuable contribution in a form of shared research work, paper reviews and suggestions that helped to present the key important findings and observations in a more meaningful way. In addition, I am very grateful to Sarah Renée Ruepp, Michael Stübert Berger and Henrik Wessing for reviewing parts of this thesis.

What is more, the experimental research activities, conducted as a part of this project, wouldn't be possible without good logistics support and acquisition of necessary equipment, facilitated by my group leader, Lars Dittmann. Thus, I am very thankful to him.

I would like to particularly thank my friend and former colleague Andrea Marcano for insightful and meaningful conversations about various work and life matters as well as great friendship during this period. I would additionally like to thank my friend Artem Shikin for good time and interesting activities outside of the busy work hours and for not letting me forget my native language. I am grateful to my colleague Eder Ollora Zaballa for giving me a late night ride home multiple times during the busiest periods of my work-life when the things were not going as smoothly as I would like.

Besides, I would like to thank all other colleagues from the Networks Technology and Service Platforms group - without them this period of my life would be less meaningful and productive. Specifically, I would like to thank Matteo Artuso and Angelos Mimidis for interesting discussions, great sense of humor and supportive attitude. Also, I would like to thank my colleagues Justas Poderys, Jakob Thrane, Jahanzeb Farooq, Line Pyndt Hansen, Krzysztof Malarski, Zifan Zhou and a former colleague Valerija Kamchevska for

being around and making my stay here more enjoyable.

Finally yet importantly, I would like to thank my family, external friends, and especially my wonderful girlfriend for her endless love, patience, encouragement and support throughout the second half of this journey.

Kgs. Lyngby, Denmark

30 April, 2018

Part I

Research conceptualization and findings

CHAPTER 1

Introduction

1.1 Motivation: the need for speed

Since the early days of the Internet, which evolved from a closed test research network, called ***ARPANET***, developed by the ***Advanced Research Projects Agency (ARPA)*** of the U.S. Department of Defense in 1969 [1], until present day, when a globally distributed communication network infrastructure is spanning the whole world, providing ubiquitous network connectivity almost anywhere and anytime, the business activities as well as private and public interactions have gradually shifted to a digital realm. Nowadays, it is almost impossible to imagine our everyday life without such activities as checking the latest news online, paying the bills via a mobile app, sharing memorable life moments with friends in the social networks, ordering goods or accessing favorite music or video content online, just to name a few.

According to the latest Internet connectivity reports from Akamai [2], the penetration of global Internet connectivity continues to grow and in the first quarter of 2017, there were 814 million unique IPv4 addresses from 239 unique countries/regions, connected to the Akamai's measurement platforms, indicating 0.7% growth as compared to the results a year prior. Global IPv6 adoption is also on the rise [2], with projected increase from 5 billion (as of 2016) to 16 billion (by 2021) of IPv6-capable fixed and mobile devices. On the other hand, according to the Internet World Stats portal [3], as of December 2017, there were approximately ***4,156,932,140*** Internet users (estimated 54.4% of world's population). As regards to the Internet connection speeds, available statistical data differs mostly due to the measurement scope/reach of a particular network connectivity provider, but the general trend shows (see Figure 1.1, based on data from [4]) that the global average Internet connectivity speeds are projected to increase by nearly doubling (from 27.5 to 53.0 Mbit/s) in the Fixed Broadband segment, nearly tripling in the mobile broadband segment (from 6.8 to 20.4 Mbit/s) with 24% ***Compound Annual Growth Rate (CAGR)*** and increasing almost twofold (from 18.2 to 37.1 Mbit/s) in the Wi-Fi connectivity segment. However, these values reflect the global average estimations, but the real situation may vary significantly from region to region, and country to country, depending on the rate of network upgrades and adoption of new innovative technologies. Thus, there is no straightforward universal mechanism for accurate extrapolation of the distribution of the Internet speeds and networked devices, even though some attempts to model the evolution of Internet broadband speeds were made by J. Nielsen [5], and are widely referred to by the Telecommunication industry as "Nielsen's Law of Internet bandwidth"

[6][7]. This engineering observation states that the ***high-end user's Internet connection speed increases by 50% per year*** (CAGR) [5], as shown in Figure 1.2 (based on data from [5]), resulting in 10% slower growth than the Moore's Law for processing power (computer processing capabilities double nearly every 18 months) [5][8], which means that the available network bandwidth plays a critical role in how fast specific categories of services are adopted and delivered at an acceptable ***Quality of Service (QoS)***.

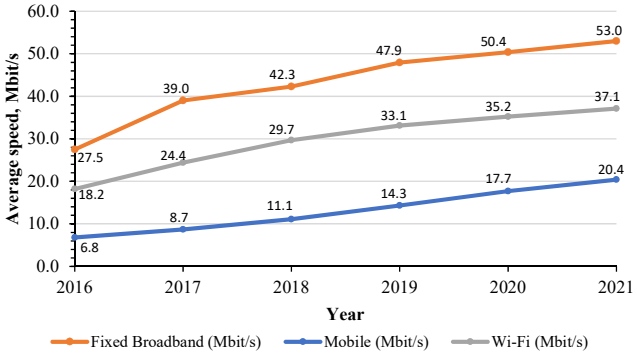


Figure 1.1: Projection of the evolution of the global average network connection speeds for Fixed Broadband, Mobile and Wi-Fi

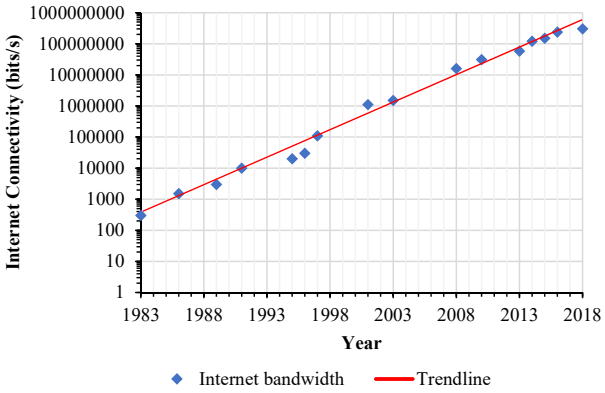


Figure 1.2: Nielsen's Law of Internet bandwidth

In reality, these projections indicate a general trend, which may be well adjusted by faster adoption and deployment of various promising technologies and connectivity

solutions, such as ***FTTx***, including ***Fiber To The Home (FTTH)***, new high-speed ***Digital Subscriber Line (DSL)*** standards (e.g., G.Fast [9]), 4G wireless access connectivity solutions (e.g., mobile WiMAX 2 [10], Long Term Evolution (LTE)/LTE-Advanced [11]), as well as standardization efforts to define 5G networking principles [12] and next-generation Wi-Fi-based solutions [13][14].

Along with the improving network connectivity and expanding Internet user base, there is an accelerating trend of nearly-exponential increase of Internet traffic [3][15]. As depicted in Figure 1.3 (based on data provided in [4]), global ***Internet Protocol (IP)*** traffic in the Internet is expected to grow almost threefold within the 5-year period (2016-2021), reaching ***3.3 Zetta Bytes (ZB)*** per year by 2021 [4]. The main contributors are IP-based Internet services (Consumer Internet traffic), such as Internet video (e.g., live streaming, Video-on-Demand (VoD)), which may reach 82% of all IP traffic by 2021, followed by general web/e-mail/data traffic, online gaming and file sharing, as it can be seen in Figure 1.4 (based on data from [4]). Mobile data traffic exibits the fastest growth rate (46% CAGR) with expected 6.7-fold increase in the period of 2016-2021 [16]. Apart from that, ***Data Center Networks (DCNs)*** are introducing another dimension of traffic growth and processing demands, since intra-DCN (originating and terminating within a Data Center (DC)) traffic constitutes a dominant portion (75.4% as of 2016, around 71.5% by 2021 [17]) of a global DC traffic volume, which is projected to reach ***20.6 ZB/year*** (94% of Cloud traffic) [17], as shown in Figure 1.5 (based on data from [17]), and this is nearly 6 times as much as the Internet-facing traffic volume. As a result, the consequence of this is twofold. On one hand side, digital transformation brings new opportunities for innovation and shorter time-to-market for applications and services, and ubiquitous network connectivity allows accessing a broad range of diverse digital content. On the other hand, there is a continuous need for operational optimization of the deployed network infrastructures and communication protocols, as well as development of new communication standards and technologies, which would allow building high performance, more agile and scalable network architectures, capable to support the increasing communication, processing and storage demands, which traditional network architectures were not initially designed for.

The main message that follows from these observations and trends is that new communication network technologies are being developed and deployed to address two sides of a global challenge: to enable the "consumption" and active usage of new feature-rich applications and services on the end-user's side (access or last-mile network segment), as well as to catch up with the tremendous ongoing growth of globally generated traffic, traversing the heavily loaded metro [18], backbone [4][19] and Data Center Network (DCN) [17] segments of the network infrastructures.

One of the key challenges, associated with such unprecedented and unbalanced traffic growth, is not only the volume of produced IP traffic, but rather the emergence of more complex and bursty traffic profiles, due to a fast proliferation of relatively new application and service models, such as Cloud service models (e.g, Software-as-a-Service (SaaS)), multi-tier web applications (e.g., a web front-end, business logic back-end, and a database

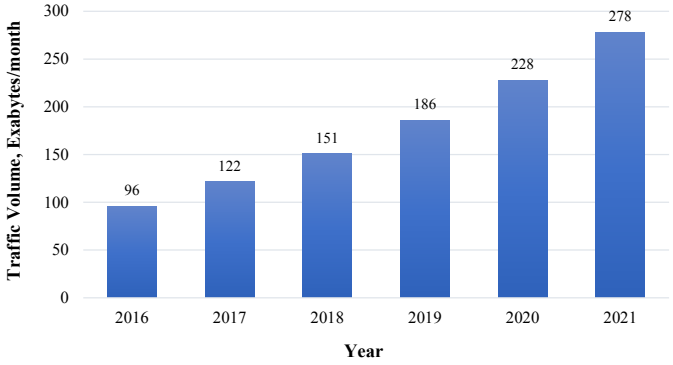


Figure 1.3: Global IP traffic growth forecast

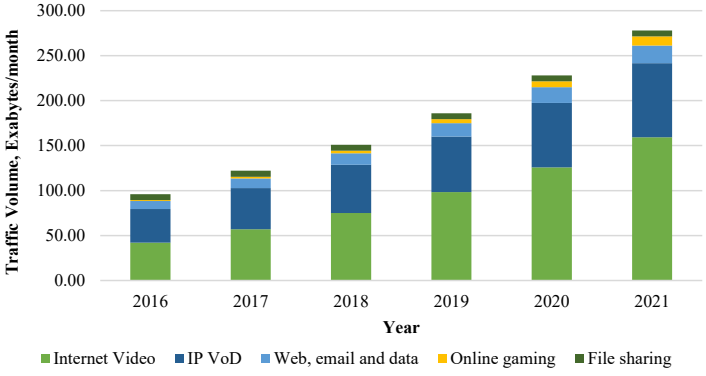


Figure 1.4: Global IP traffic growth forecast by application

component) and parallel/distributed data processing and application programming models (e.g., scatter-gather patterns of Web search services, MapReduce [20] and Big Data Analytics) with very stringent timing and bandwidth requirements in DCs. In addition, since the Internet-based video streaming services (YouTube, Netflix, Hulu, Facebook Live, etc.) currently constitute the largest portion of global Internet IP traffic, which is projected to increase significantly, traffic patterns shift from steady/constant to more dynamic [17], and new multimedia-related services are rapidly appearing as a result of wide adoption of open Application Programming Interfaces (APIs) (Facebook, YouTube, Twitter, Skype, etc.) [17]. As a result, the peak-hour (the busiest 60 minutes of a day) Internet traffic is projected to increase by a factor of 4.6 in the period between 2016 and 2021 [4], and grow faster (35% CAGR) than the Average Internet traffic (26% CAGR) [17].

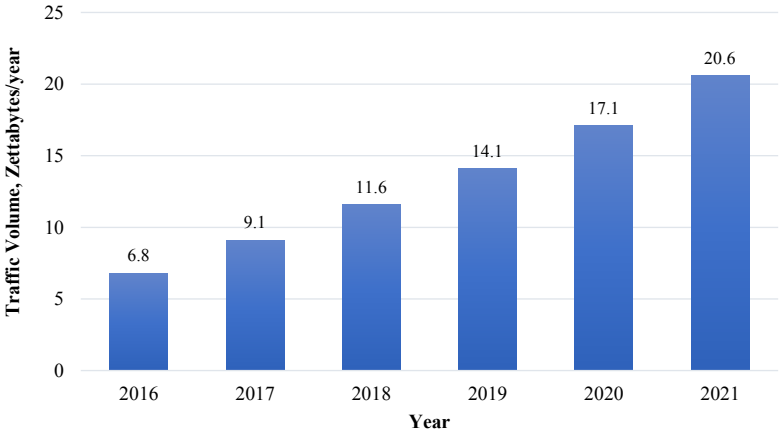


Figure 1.5: Global Data Center IP traffic growth forecast

The conclusions that can be drawn from this discussion, are as follows:

- The effect of digital globalization in a form of massive increase of the Internet user base, gradual increase of the Internet connectivity speeds, new feature-rich applications and services, the rise of critical importance of the ***Information and Communication Technology (ICT)*** in business growth and optimization processes, have resulted in tremendous increase of global IP-based traffic volumes, which traditional network design and operational practices, as well as communication protocols, were not ready to handle.
- New communication network architectures, technologies, protocols and algorithms, both standardized and best-practice solutions, including Cloud and virtualization technologies, have been developed and deployed to tackle the arising performance, efficiency and scalability challenges. However, the increasing scale of networks and provided services, as well as increasing degree of network softwarization, is shifting the degree of complexity from hardware to software domain, and, as a result there is a strong need for automation of these activities to reduce the operational and management complexity. In theory, such capabilities are offered by the Cloud [21] and ***Software Defined Networking (SDN)*** [22] paradigms and a range of related platforms, tools and technologies, such as ***Development and Operations (DevOps)*** [23] framework of concepts and practices for automation of IT processes in software engineering. In practice, network operators and IT professionals report that the key challenges and concerns being faced in real-life deployments are as follows: testing the software interoperability [24], security and monitoring performance issues [24][25], as well as integration with DevOps principles [25].
- A "one-size-fits-all" model of network architecture design and deployment is not

applicable for all the diverse network environments due to fundamentally different performance requirements for different types of networks, such as Wide Area Network (WAN)/transport or DCNs. Thus, custom-built solutions may be the most viable option for large-scale enterprises and operators with already established operational practices, in order to be able to seamlessly integrate new software-defined network transformation strategies with existing legacy deployments, while small-to-medium enterprise sector may benefit from available pre-designed vendor-specific products.

- Interoperability testing and integration, as well as performance monitoring and analysis remain some of the key critical aspects for successful adoption, practical deployment and operation of next-generation intelligent, automated, software-defined and scalable network architectures. Moreover, associated traditional and new communication protocols and extensions, which must co-exist together, have a strong impact on the operational stability of heterogeneous networks and acceptable consumer- and business-level service experience. Therefore, practical deployment/integration complexity, associated investment costs (new network gear, competence acquisition/staff training, etc.) and risks (unclear business impact/benefits of a new disruptive solution), mandate a comprehensive and multi-layer (top-down or bottom-up approach) testing and performance evaluation of new technologies and innovative solutions, where testing scale, testing performance and accuracy are the definitive factors, which may affect the decision-making process.

1.2 High Speed Network Performance Evaluation: challenges, solutions and open research issues

As it has been pointed out in Section 1.1, global IP traffic sets record high growth rates, and this trend is predicted to continue. In this context, it is important to highlight two aspects, which are directly affected by such massive traffic increase, namely the ***network congestion*** state and ***network performance evaluation and testing***. The former aspect becomes much more difficult to control due to rapidly changing traffic profiles and variable daily and weekly service usage patterns, as well as increasing levels of traffic aggregation/density in metro/core and DC networks [18]. In the latter, it is becoming more difficult to model and test large-scale communication systems and services as well as to collect and process large volumes of measurement data and use this data for timely network performance optimization. These two aspects are discussed in Sub-sections 1.2.1 and 1.2.3, respectively. Common network testing approaches are summarized in Sub-section 1.2.2. These observations form the background for a discussion of open research questions, which motivated this research work and will be outlined in Sub-section 1.2.3.

1.2.1 Network congestion in a nutshell

It is important to emphasize the impact of continuous nearly exponential traffic growth on the operational state of the deployed networks, with a primary focus on ***Network Congestion***, as this has become an area of concern and active ongoing debates for the ***Internet Service Providers (ISPs)*** and ***Content Service Providers (CSPs)*** [26][27], especially in the light of debates around ***Net Neutrality*** [28][29] and its possible consequences for network operators.

A ***Network Congestion*** characterizes the state of a network, when a particular network output link, a node (e.g., switch or router, etc.) or an entire network segment is overloaded with excessive amount of incoming data, exceeding the available capacity, and the network is not capable of providing acceptable ***Quality of Service (QoS)*** [30][31][32]. This condition is usually accompanied by increased queueing delays, packet/frame loss and reduced network throughput. Hence, in this case, ongoing increase of the input traffic load leads to proportionally smaller increase, or even decrease of throughput. In the context of end-to-end communication between a service host (e.g., a server) and service consumer (e.g., a client terminal), the global communication network appears to be a "black box" with variable properties, which are difficult to control and, especially, to predict. A high-level view of this situation is depicted in Fig. 1.6.

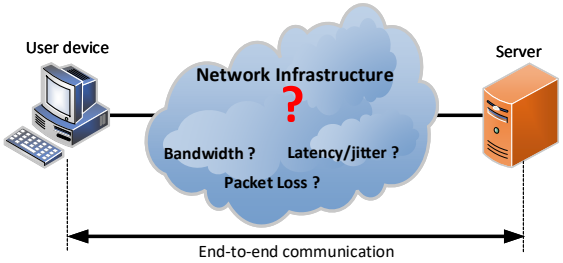


Figure 1.6: End-to-end communication: service delivery challenges

The fundamental reason leading to such detrimental networking effects is generally limited network resources, such as packet processing capacity of a node or link speeds. However, the problem is much more complex than just processing and storage capabilities of a network device. Hereby, a summary of other key important and, actually, definitive factors, is outlined as follows:

1. ***Bandwidth oversubscription***. This is a commonly used strategy of network design among network operators, which incorporates the statistical bandwidth usage information (e.g., collected by monitoring, analyzing historical usage data) into the architectural network design decisions [33][34] by applying ***Bandwidth Oversubscription Ratio (BOR)***. The main idea is that, considering that most of the network design practices follow layered/hierarchical network structures with access/edge,

aggregation/distribution and core layers or 2-tier designs without aggregation, the sum of the total provided uplink bandwidth is not equal to the total bandwidth of all the downlink links of a considered network layer. This difference is defined by a chosen ***BOR*** for each inter-layer (access-aggregation, aggregation-core, spine-leaf, etc.), increasing in the top-down direction in the network hierarchy [33][34]. This is performed in order to still be able to provide reasonable QoS for the deployed services while avoiding the significant increase of the network deployment costs. The impact of this factor is such that, this strategy allows handling average traffic loads well, since, statistically, network resources are known to be underutilized, but improperly set BOR values (e.g. too high) may inflict degraded performance of certain types of services at peak hours due to a congestion.

2. ***Configuration of network protocols***. Such factors as routing misconfiguration at the network layer may result in undesired performance degradation effects, when a congested or longer path or slower link is being chosen to re-route the traffic in case of a failure in the network, exacerbating the congestion state even more. This applies to layer 2 protocols as well (e.g., Spanning Tree Protocol (STP)).

3. ***Specifics of Transport layer (4) protocols***. Algorithmic properties of widely deployed network transport protocols (layer 4 of Open Systems Interconnection (OSI) model), such as Transmission Control Protocol (TCP) [35] and User Datagram Protocol (UDP) [36], present another challenge for the stability and performance of communication networks. The reason is that TCP, being a reliable and connection-oriented protocol, relies on a variety of congestion control and loss recovery algorithms, commonly used together with the generic TCP, and tries to retransmit any data, potentially lost due to a congestion. The problem is that, depending on the severity of ongoing network congestion, excessive retransmission of packets may exacerbate the current congestion state even more, possibly leading to prolonged congestion and service disruption times for other flows. The exact "response function" depends on the types and combination of different enabled algorithms within TCP. On the other hand, UDP streams are connection-less and without any rate control, and certain UDP-based applications (e.g., video streaming) may exhaust the available shared bandwidth, degrading the performance of TCP flows, and possibly creating a new network congestion. In addition, the per-service traffic load scale at particular time of the day/week and specific usage profiles, e.g., Internet-based video downloading (pre-fetching) services like Netflix or YouTube (both use TCP) are reported to account for more than 40% of download traffic at peak times in the US [30][37]. That creates a massive load on the network at the peering points among ISPs and Content Delivery Network (CDN) providers, causing transient network congestion events.

4. ***Multi-tier services/applications***. New feature-rich, distributed applications and services, such as Web search, social networking or other multi-layer applications,

are widely exploiting parallel processing techniques and produce scatter-gather traffic profiles [38][39], which produce "burstiness" of traffic, and are known to create such widely known performance degradation/network congestion effects as TCP Incast or Outcast [40][41], when multiple components (workers) of a distributed application, generate responses to a processed request at near the same time, destined to an upper tier aggregator, and the uplink may become congested.

5. ***Network Heterogeneity and scale***. Almost ubiquitous network connectivity, different types of access technologies deployed (fixed broadband, mobile, Wi-Fi, satellite) as well as such actively researched and promoted architectural network design approach as ***Fixed-Mobile Convergence (FMC)*** [42][43], which defines new deployment models of mixed wired-wireless network architectures, create new challenges in terms of protocol design, requiring new adaptive algorithms and protocols, which can provide seamless communication experience for different types of transported data flows over mixed communication mediums with varying transmission properties. Current state implementations, including TCP protocol suite, are reaching their limits and require new design considerations, as well as additional performance evaluation and testing. Increasing network scale requires a very well-thought-through network transformation and upgrade strategy in a long run, since, e.g., scaling the network in a hierarchical-vertical manner leads to longer communication paths (i.e., larger number of hops) between distributed components of a DC, introducing more potential network hot-spots and increasing the probability of congestion.

A mechanism worth mentioning and widely used by the ISPs to manage the traffic, which is known as traffic "throttling" or capping [44][45][46], in which, by using various traffic policing, shaping and inspection mechanisms (e.g., Deep Packet Inspection (DPI)), ISPs are able to create artificial "traffic congestion" for particular flows (e.g., Peer-to-Peer (P2P) traffic) or even services (e.g. video streaming), which may be overloading the network. Hence, this is a controversial network protection mechanism on one hand side, and an element of unfair competitive advantage from the network operators' perspective.

The complexity of end-to-end service performance and congestion management is illustrated in Fig. 1.7, which shows that a typical communication network path used to deliver a requested service may be composed of a large number of intermediate network elements and entire network segments with diverse performance characteristics and requirements, increasing the probability of encountering multiple network congestion hot-spots. To alleviate the severity of possible congestion and to derive more efficient network design and operation strategies, comprehensive network testing and performance evaluation approaches are of paramount importance. This aspect will be detailed in the next Sub-section.

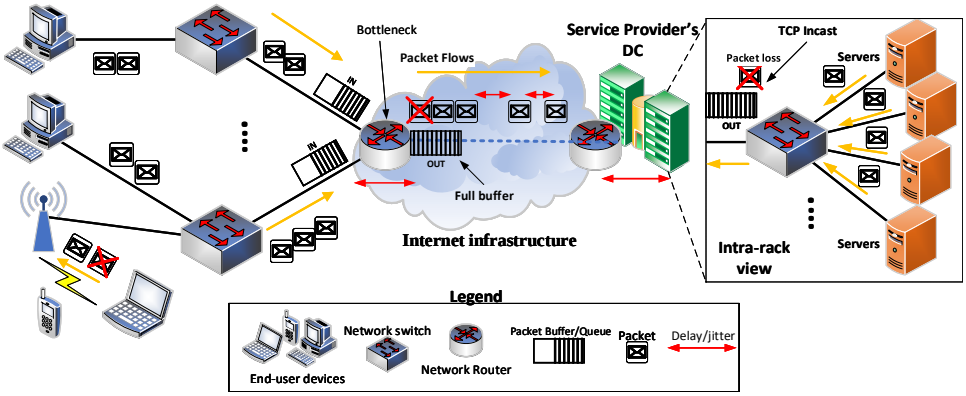


Figure 1.7: Network congestion: end-to-end perspective

1.2.2 Common practice in network testing

There are generally two main functional components of network performance evaluation and testing process, such as: 1) specification of testing objectives and metrics to evaluate (***what to test***); 2) identification, design and implementation or adaptation of an existing testing/measurement methodology and tools to use (***how to test***).

A widely used network test setup configuration is depicted in Fig. 1.8, where performance benchmarking experiment for a Service/Device/Network Under Test can be carried out by connecting the corresponding output/input of a network tester device (e.g., traffic generator's interface) to an input/output of a system, which needs to be tested. The exact physical connectivity (the number of interfaces/test links) depends on the specifics of a particular test (e.g., as defined in respective benchmarking specifications [47][48]) and handbooks [49].

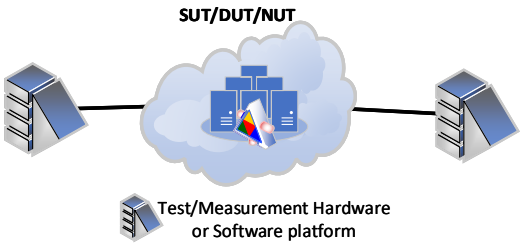


Figure 1.8: Common testing configuration. Note: SUT (Service Under Test), DUT (Device Under Test), NUT (Network Under Test)

In practice, the choice of a particular testing methodology greatly depends on the pursued goals, required accuracy and measurement scale, as well as the availability of test

equipment and software. Therefore, network performance evaluation methodologies can be categorized as follows:

1. ***Network Simulation/Emulation in software***. This is one of the most widely used methods, allowing to model almost any network topology or configuration in a specialized (e.g., optical [50], SDN [51] and DC [52] or Cloud [53]) or multi-functional network simulation tools, such as in [54][55][56] and their extensions, which can be open source or commercial products with associated pros and cons.

2. ***Small-scale physical-only lab testbeds***. As reported in [57], as of 2013, most of the published SIGCOMM papers on network performance testing use setups with, on average, 5-8 network switches and 10-15 servers.

3. ***Large-scale physical network testbeds***. In order to be able to conduct quality research on the challenges present in real large-scale communication networks, it is important to have access to scalable test resources, preferably within comparable order of magnitude. Such testbed environments can be grouped into:

    - ***Distributed large-scale testbeds***, such as PlanetLab [58], OneLab [59], GEANT [60], M-Lab [61], etc., designed for global research and education purposes.
    - ***Specific research-project-based testbeds***, such as OpenStackEmu [62], GENI [63], FEDERICA [64] and other related projects [64]. These testbeds target distributed testing, WAN environments and global resource consolidation.

4. ***Hybrid physical-emulated/simulated testbeds***. This is a relatively new form of a network testing environment, allowing to interconnect real physical and simulated/emulated devices into a unified testbed. OpenStackEmu [62] belongs to this category as well.

5. ***Analytic-mathematical modelling***. This form of network performance modelling is not widely used for large-scale network performance evaluation due to the associated complexity and high level of abstraction of the modelled processes, which makes it more difficult to apply in practice. However, this form of modelling is still very useful for performance modelling of communication protocols or algorithmic features, using the elements of queueuing networks theory [65], whereas the problem of network scale in modelling can be approached with relatively new network decomposition methods (e.g., [66]), which can be used to scale the analytical models, applicable to larger network models.

One of the most important components of a network test/benchmarking setup (i.e., a testbed) is a traffic load generator, which would be capable of creating different traffic profiles and possibly mimicking the realistic communication patterns of real networks, so that accurate and more realistic network stress-testing conditions can be replicated to

enable comprehensive network performance evaluation tests. Existing traffic generation mechanisms can be grouped in the following manner:

1. ***Software-based traffic generators***, such as Ostinato [67] or hping3 [68].

2. ***Simulated traffic sources***, which produce synthetic traffic profiles in a simulation environment, where, in reality, no real application data is generated, but rather in-memory data structures with packet header information and a traffic pattern, mimicking a specific traffic source (e.g., video streaming, file transfer, etc.).

3. ***Hardware-based traffic generators***. This category represents advanced, powerful hardware platforms with specialized network testing software, including traffic generation tools, fine-grained measurement/analytics capabilities, as well as entire network/technology and protocol-specific test suites, e.g., such as a state-of-the-art large-scale Cloud DC stress-testing framework in [69]. Other examples include solutions offered by Xena Networks [70], Spirent [71] and Ixia [72].

4. ***Real pre-recorded traffic traces replayed though software***. This is a specific case, where pre-recorded traces of real network communication sessions can be replayed in software to synthetically reproduce a realistic traffic profile. This method is often coupled with simulation tools (e.g., [53]) or supported by hardware-based testers.

The following Sub-section discusses the challenges associated with large-scale network testing and performance measurements in the context of the previous discussion, and outlines open research questions, which need additional consideration.

1.2.3 Open research questions in network performance measurement and testing

Network testing at large scale has been a major issue in communication networks' research for many years. The methodologies and solutions, discussed in Sub-section 1.2.2, have both use-case-specific advantages as well as various limitations, as it was pointed out earlier. However, in the context of large-scale/high-performance network testing, the requirements are much tighter, and the applied testing methodology may require a combination of several different approaches together to obtain any reasonable results. The main network performance evaluation challenges, restricting the capabilities of communication network researchers and test engineers, are as follows:

- There is no universal "affordable" solution (from a financial, footprint and complexity points of view), which would satisfy such contrary requirements as testbed's scalability, high performance and testing/measurement accuracy, fully-featured test services and applications (as found in production environments) and reasonable implementation/deployment costs being the most definitive factor.
- The following considerations are of concern in the scalable network testing context:

  - In the case of ***distributed large-scale testbeds***, mentioned earlier (PlanetLab, OneLab, etc.), we can accomplish the goal of creating a large-scale distributed testbed by, e.g., setting up a virtual overlay network running on top of this infrastructure, where we can use the software-based switches and routers to form a test network of interest, as well as run test applications on Linux-based hosts to generate real traffic flows. The problem in this case is that we cannot control or manipulate the configuration and processing parameters of the underlying physical infrastructure, as well as we cannot control the intermediate network connections, since this is a distributed Internet-based infrastructure. Other possible solutions include using paid Amazon Virtual Private Cloud (VPC) services [73] with full control of a virtual slice on the network with guaranteed QoS, but again we are only in control of a logically isolated virtual infrastructure, and the requested Cloud network scale defines the cost of a requested service.
  - Simulation as well as an emulation approach is suitable for protocol-algorithmic level studies, due to the following reasons: a) ability to focus on low-level protocol details and properties in a controlled and isolated environment; b) allows for defining and running repeatable experiments with reproducible results, useful when, e.g., the research objective is to evaluate the impact of a specific parameter or algorithmic change on the protocol behaviour. On the other hand, a simulation environment abstracts the real network processing effects and physical properties, which may be crucial to test (e.g., optical switching latency) or real application layer behaviour (e.g., response times).
  - As regards to small-scale physical testbeds, they are useful for simple experimental tasks, such as testing network failover or local load-balancing algorithms in simple configurations, and allow observing some network-level processing effects, but such setups cannot be really used to demonstrate or study, e.g., network power consumption or optical switching gains (fast optical path versus slow electrical path) or other scenarios. This is a critical factor, affecting the accuracy and applicability of research results at large scale.

- Another critically important aspect of large-scale/high performance network testing is related to collection, storage and analysis of large datasets obtained by measurements. This aspect requires consideration of new data collection, distribution and processing architectures, combination and customization of different tools and platforms in order to create a flexible and manageable high performance data analysis environment [74]. Therefore, this is a highly practical aspect of network testing and efficient data distribution, e.g., vitally important for large-scale scientific research experiments, such as Large Hadron Collider experiments at CERN [75], producing enormous amount of measurement data (tens of GB/s), which needs a complex and highly optimized network infrastructure to collect and distribute such volumes of Big Data.

- Other critical factors are traffic generation methods (described earlier), where accuracy of performed measurements (e.g., due to clock synchronization issues like clock drift/skew, timing/clock granularity on the underlying OS used, etc.) as well as lack of full-featured and functional implementations of realistic multi-tier Internet applications and services available for testing and modifications are potential issues.

The discussion, presented in this Sub-section, pointed out a large number of open research questions, which require additional investigation or new outlook/solutions to explore. This Ph.D. thesis is an outcome of a Ph.D. research project, which was composed of research activities, conducted in close collaboration with two other research projects, namely: a national "Layer 4-7 testing at 100 Gbps" and an EU FP7 "***Combining Optics and SDN In next Generation data centre Networks (COSIGN)***" project. Therefore, the following open research questions were investigated in this work, both in relation to the aforementioned projects, as well as additional undertaken research initiatives:

1. In the context of Layer 4-7 testing: analysis of the protocol-algorithmic properties of high speed ***TCP Congestion Control (CC)*** aspects, focusing on the currently globally deployed TCP ***CUBIC*** extension. The main properties under investigation are the algorithmic stability and loss recovery efficiency of high-speed TCP CUBIC connections in high-speed networks (e.g. the Internet). The reason why a Transport layer (4) has been targeted within the Layer 4-7 context is that a functional, stable and scalable TCP connection engine and supporting algorithms build a foundation for reliable and adaptive application/service delivery in modern network environments. Thus, any malfunctioning or miscofiguration of the Transport layer may lead to severe performance degradation for all the upper layer protocols, and there is still a broad range of associated problems, which must be tackled. Hence, in this work, only a subset of open research issues in TCP CC has been investigated. This evaluation does not cover mobile/wireless network environments and DCNs.

2. In the context of COSIGN data center research: practical-experimental work on implementation and testing of a hybrid physical-simulated testbed for scalable ***Data Center Network (DCN)*** performance characterization and testing has been carried out. The main focus areas are: scalability of a DCN testbed and ***Integrated System Test (IST)*** approach of DCN testing.

3. Additional research probes include a study of energy efficiency in DCNs, testing and analysis of a Flow rule placement mechanism for SDN switches, and analysis of SDN Traffic Engineering (TE) capabilities for DCNs.

1.3 Research contributions

The research work contributions, presented in this thesis, as stated earlier, are comprised of two parts in relation to two research projects, as well as additional efforts in the areas

of energy efficiency in DCNs and Flow rule placement algorithm performance studies as well as an analysis of SDN-based TE for DCNs. The main contributions of this work are summarized in Figure 1.9.

Layer 4-7 testing

Analysis of algorithmic robustness of high-speed TCP CUBIC connections: A customized simulation model has been created in Riverbed modeler, containing additional algorithms and fixes, based on the latest updated Internet standards (RFCs) to improve the stability of TCP Congestion Control (CC) mechanisms (Reno, NewReno, CUBIC) of native implementation of TCP module in the modeler; the algorithmic changes are implemented in C/proto-C language and validated through extensive testing. The simulation model is used for the analysis of algorithmic properties of high-speed TCP CUBIC connections with large congestion windows in communication corner-cases, which have limited exposure in the literature, in order to complement the insight of the adaptation capabilities of current default TCP CUBIC CC algorithm (used in Linux and Windows 10 TCP stacks) in difficult communication conditions.

Analysis of packet loss recovery efficiency of TCP CUBIC connections: the customized simulation model is extended with extra traffic sources and algorithms for better-informed packet loss detection and recovery, such as Limited Transmit (LT), ACK Heuristics, conservative SACK-based recovery (RFC 6675) and Proportional Rate Reduction (Conservative Reduction Bound (CRB) and Slow Start Reduction Bound (SSRB) versions). This updated model is used to evaluate the Packet Loss Recovery Efficiency (PLRE), as a compound metric, of high-speed long-lived TCP CUBIC connections. The obtained results allow for better understanding of how different combinations of loss detection/recovery algorithms affect the stability, convergence speed and, ultimately, flow completion times of high-speed TCP CUBIC-based connections.

A survey-analysis of Congestion Control aspects in high-speed Internet: a comprehensive study of the evolution of TCP CC mechanisms from the first attempts to regulate the network congestion to current state-of-the-art and experimental solutions, their associated pros and cons, non-TCP-based CC frameworks, as well as open research issues, requiring additional studies. This study will help to form a better insight of the complexity of CC in modern high-speed network environments, such as the Internet.

DCN Performance characterization and testing

A Proof-of-Concept setup of a hybrid testbed for DCN testing: an initial experimental test setup was assembled, comprised of a Simulation framework (Riverbed modeler) with a System-in-the-Loop (SITL) module for real-time simulation functionality, two modeled DCN structures (a Hypercube and a Ring-of-Rings), a set of high-performance hardware-based network testers and portable workstations (hosting software traffic generators and the modeler) for traffic generation and performance measurements (delay, packet loss). This test setup has been used for the Proof-of-Concept (PoC) step to evaluate the feasibility of creating a hybrid (physical-simulated) testbed for DCN performance studies,

and this work has formed a foundation for the subsequent studies and gradually introduced enhancements.

Enhancement of the testbed with real DCN switches: the PoC experimental setup was transformed into a more powerful testbed, where the simulation environment was migrated to a powerful modular server, and a hybrid DCN topology was formed from physical (electrical and optical DCN switches) and simulated (vendor-specific or custom switch models) network devices, by interconnecting them via the optical-electrical ports and virtual SITL interfaces of the simulation server and the network interfaces of real physical switches. The key system integration challenges faced and resolved are: configuration/adjustment of the real-time simulation and packet conversion (via SITL) parameters, alleviating the initially encountered packet loss and excessive buffering latency (on a Windows-based host). The purpose of this work was to investigate the possibilities to scale a studied DCN topology to larger dimensions, while retaining some realistic network processing effects in real network hardware.

Integration of SDN control framework with the testbed: the testbed was further enhanced by integrating an external SDN controller (OpenDaylight Lithium and Boron), running on a separate physical server, with this hybrid setup. This activity involved setup and configuration of the controller, OpenFlow (OF) agents in the physical switches (electrical SDN DCN switches, Optical Circuit Switch) and OF process model of the simulated SDN switches, connectivity (control and data planes) testing. The configuration of the simulated SDN switches was successfully performed via the REST interface and OF control channel of the SDN controller. This step allowed adding more advanced SDN-based, unified control and configuration capabilities to this testbed. The ultimate benefit is that the reconfiguration of the topology or a specific device can be automated and achieved in a centralized manner via the controller.

Performance evaluation and fine-tuning of the testbed: initially, the main detected limitation of the testbed was that the entire simulation model was running in a single thread of one logical CPU core, which was significantly constraining the real-time packet processing capabilities of the simulation model. The key performance tuning activities carried out: parallel packet processing on a multi-core system, kernel-level and Network Interface Card settings' adjustment, multiple tested OS-level optimized performance profiles (latency, throughput, etc.), as well as using a Real-Time OS kernel with preemption, and other adjustments. This extensive set of tests allowed identifying the maximum achievable throughput (Mbit/s and packets per second) at which the system shows relatively stable performance (end-to-end latency and jitter, memory usage, packet buffering) in different configurations (the number of modeled network nodes) in its current state.

Analysis of the impact of WDM-enhanced optical switching in multiple DCN architectures on the power consumption: energy efficiency as a performance criteria of DCNs is explored, and the outcome of this work shows the impact of optical switching, enhanced by Wavelength Division Multiplexing, applied at different layers (core, aggregation, access) of several DCN topologies, on the power consumption level. Riverbed's SP Guru Transport Planner software was used to perform DCN dimensioning, and these results

were used as an input to a derived power consumption calculation model.

Experimental evaluation/analysis of a Flow rule placement algorithm in DCN SDN switches: a new proposed Flow rule placement algorithm, implemented in ONOS SDN controller, exploits the hybrid Flow tables (composed of hardware and software implementations) of DCN SDN switches under test. Performed experimental testing and analysis revealed potential service performance implications of the Flow rule migration process (between the hardware and software tables), with a trade-off between the number of unique accommodated flows and latency variance for in-software processed packets.

A short survey-style analysis of SDN TE capabilities for DCNs: this study is intended to provide a condensed yet comprehensive and critical analysis of the benefits of SDN-based testing of various Traffic Engineering (TE) approaches for DCNs. Provided examples of a test use-case and obtained experimental and modelling results highlight the potential of applying a hybrid physical-simulated DCN testbed to achieve the testing scalability goal.

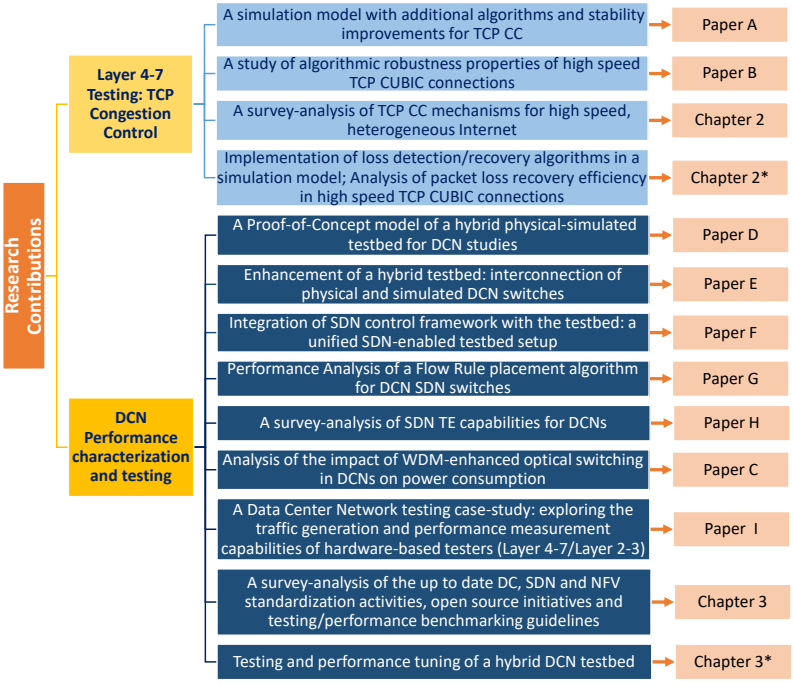


Figure 1.9: Documented Research contributions and additional research initiatives. Note: * - a related paper is under preparation

1.4 Summary of the related papers

This section presents a short summary of the related publications, included in this PhD Thesis, highlighting the research problem, the methodology used as well as the key research findings and results.

Paper A/Poster

High Performance Network Evaluation and Testing

In this work, presented as a poster, a simulation-based study of high-speed TCP CUBIC connections is conducted in three distinct scenarios, targeting communication corner-cases, which have limited exposure in the literature. We assess performance and operational stability of high speed TCP CUBIC algorithm in a network environment with multiple long-lived high-speed (CUBIC) and regular (Reno, NewReno) TCP connections with large congestion window sizes, simultaneously sharing an oversubscribed network link under variable communication conditions (packet loss, increasing RTT, variable network buffer sizes). A dumb-bell network topology is used in all the scenarios in a packet-level Riverbed modeller with additionally implemented algorithms (Appropriate Byte Counting) and fixes (Slow Start Threshold adjustment, CUBIC window growth fix) to improve the stability of the modelled TCP connections. The implemented traffic generation method ensures that TCP connections always have new data to send. The obtained results indicate that high speed TCP CUBIC connections are able to obtain a significantly larger bandwidth share (as compared to TCP Reno/NewReno) in the low-to-medium random packet loss region and switch to the TCP-friendly mode with comparable performance under high packet loss even under increasing RTT of the flows. Buffer sizing of the network nodes allows avoiding excessive burst packet losses (small buffer regime, 25-50% of a Bandwidth-Delay Product (BDP)) at the cost of limited throughput, but the network buffer (at the bottleneck link) occupancy is almost 100% most of the time under different sizing regimes.

An additional aspect, highlighted in this work, is related to large scale network testing possibilities and limitations. Network testing at scale is a complex matter, especially considering algorithmic scalability and stability properties of a large number of concurrent high speed flows, and simulation environment might constrain the obtained results and limit their applicability in realistic network setups. Hence, flow emulation using a high performance hardware platform capable to handle a large number of flows (e.g., millions) can be a viable approach, but the open research questions are how to limit the overhead of TCP connection state and statistical information to accommodate a large number of high speed flows in a resource-efficient manner.

Paper B

Robustness of Multiple High Speed TCP CUBIC Connections Under Severe Operating Conditions

This work is a continuation of the work, presented in ***Paper A/Poster***, and extends the simulation-based analysis with additional varied parameters and performance metrics in a scenario with increasing random packet loss pattern, increasing RTT of the established connections, added mixed background traffic sources (with Poisson and Pareto arrival processes), and provides more indicative results to better understand the evolution of different performance parameters of high speed TCP CUBIC connections in a lossy, high capacity network with large Bandwidth-Delay Product (BDP). The key performance indicators of interest in this work are the evolution of Congestion Window (CWND) size, time-average throughput and bandwidth sharing fairness indicator, namely Jain's Fairness Index (JFI), providing a rough estimate of how equally the bandwidth is shared among multiple heterogeneous long-lived (elephant) TCP flows. An important introduced condition here is that the application layer is continuously generating new data (flows always have data to send); hence, in this way, we are focusing on the flow dynamics of the TCP connections, defined by the algorithmic properties of the congestion control mechanisms used (in CUBIC, and in the traditional Reno and NewReno), and not being limited by a specific communication pattern of an application used. TCP connections, using traditional Reno and NewReno congestion control are modeled and evaluated together with CUBIC flows to have a baseline scenario for a comparison. The results show that high speed CUBIC flows experience proportional decrease in throughput with increasing RTT under moderate loss conditions (BER = 10^{-9}), whereas in high packet loss region (BER = 10^{-6}), where the TCP-friendly mode of CUBIC is activated, the algorithm is able to recover the throughput faster (in non-linear leaps) with the increase of RTT, compared to the standard TCP Reno/NewReno. The general observation is that high-speed flows with large CWND, operating under large Bandwidth-Delay product conditions, are affected more due to larger bursts of packets sent to the network within the transmission round as a result of statistical dependency of the number of randomly dropped packets on the current CWND size.

Paper C

Energy Efficiency Benefits of Introducing Optical Switching in Data Center Networks

Increasing scale of DCNs as a result of tremendous global IP traffic growth, have raised serious concerns about the power consumed by these facilities, and novel energy-efficient architectures and power management solutions are actively being investigated. This paper targets the Energy Efficiency aspect of DCNs by presenting a power consumption study in three different DCN topologies with optical switching, such as a traditional three-tier Tree, a Fat-Tree and a Ring-based structure. The main novelty of the study is that it considers the impact of selective (on particular network layers) optical circuit switching, enhanced by the Wavelength Division Multiplexing (WDM) capabilities, applied to a DCN. A

Transport network planning tool is used for network dimensioning with different degree of applied traffic grooming. The obtained dimensioning results are used as an input to a power consumption calculation model, based on the nominal power consumption values of different components (transponders, ports per node type, such as optical, electrical, electro-optical), obtained from the industrial technical specifications. The results are of indicative nature, since the modelled topologies are not large in the context of DCNs, but the key observation is that enabling optical switching only at the aggregation layer results in the highest energy savings in the Fat-Tree and the traditional Tree, while an optically switched core benefits most from the ring-based network. For the latter, the core ring nodes need fewer long-reach transponders at the trunk interfaces and benefit from more efficient traffic grooming in the access part. The study could be improved by modelling larger DCN topologies and considering the impact of different non-uniform traffic profiles (e.g., Cloud-based traffic).

Paper D

Combining Hardware and Simulation for Datacenter Scaling Studies

One of the biggest challenges in communication networks research in general, and in Data Center Network (DCN) research in particular, is very limited or no access to production scale network infrastructures for experimental and testing purposes, which is a vital component of quality research. This situation is hindering the possibilities of fast innovation and applicability and transfer of new promising research ideas from lab environments to large-scale production networks. This paper presents a promising approach, which may help addressing this problem by combining available real physical hardware with simulated DCN components, capable to communicate in real-time. This approach introduces new possibilities for performance and scalability characterization of DCNs at scale. A hybrid physical-simulated DCN testbed setup is assembled for the Proof-of-Concept tests, consisting of commercial high performance hardware traffic generators/testers and a simulation framework with a set of modelled DCN topologies (16-Hypercube and a Ring-of-Rings). This connectivity scenario emulates inter-DCN communication scenario, when a modelled DCN structure is loaded with real traffic streams, produced by external hardware-based network testers, acting as peering DCNs. The physical and simulated devices are linked via System-in-the-Loop (SITL) virtual gateway modules of the simulation tool, providing real-time communication experience between the physical and simulated components. The preliminary testing results indicate that the SITL gateway adds a conversion delay in the order of a few microseconds (4-6 μs) as well as load-dependent buffering delays that must be taken into consideration for any latency measurements. The achieved pass-through throughput of the linked system, when the simulation environment is running in a single thread, was around 80 Mbit/s and 120 Mbit/s for ICMP- and TCP-based flows, respectively. The approach presented here has a strong potential in the emerging integration of optics in the Data Center (DC) world, where scaling and latency effects must be studied without necessarily having access to real DC infrastructures.

Paper E

A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks

This paper presents a continuation of the research activities, discussed in Paper D. The challenge of scalable performance characterization and testing of DCNs is an open research question with ongoing efforts. In this work, a hybrid physical-simulated DCN test setup (Proof-of-Concept (PoC) version) is transformed into a more powerful and functional DCN testbed, comprised of physical enterprise class DCN switches, a free-space high-radix Optical Circuit Switch (Polatis) and high performance hardware-based network testers. The main goal pursued in this work is to combine physical and simulated DCN network equipment (switches) into a unified DCN structure with an ultimate goal of addressing the DCN scalability challenge via large-scale topology modelling in a real-time simulation. Therefore, in this test scenario, a 16-node Hypercube DCN structure is created by interconnecting 8 physical switches with 8 simulated switches and an OCS. Hardware network testers are used for DCN connectivity- and stress-testing. The simulation model is hosted by a powerful DC server with up to 16 electro-optical interfaces used to interconnect the physical and simulated nodes. The test setup is used to evaluate the latency reduction gains of introducing fast OCS to form optical bypass connectivity, e.g., to enable load redistribution/offloading of DCN segments with high utilization. End-to-end latency measurements and experiments show that even in a 16-node DCN structure, there is nearly 50% latency reduction when steering the traffic (e.g. elephant flows) to optical shortcut links, as compared to multi-hop electrical path. An outline of potential use-cases for this testbed is provided, forming a better insight of the applicability of such hybrid testbeds for comprehensive and scalable DCN testing.

Paper F

A Novel Hybrid Testbed Combining Optics and SDN for Topology-Agnostic Datacenter Network Evaluation

In this paper the DCN testing and performance evaluation scalability aspect is brought one step forward by attempting to address not only the scaling in terms of the number of introduced network nodes or DC servers in a DCN testbed environment, but focusing on the programmability and automation aspects. The reason is that, despite the benefits of large-scale modelling and better control of the networking parameters in the simulation environment, the complexity of network re-configuration, change of specific link or node parameters or other changes, which need to be applied to a large set of nodes (e.g., change of routing parameters/protocols), increases significantly and there is a need for automation or software-driven control of these tasks. Therefore, this paper presents enhancements of the created DCN testbed in a form of **Software Defined Networking (SDN)** functionality. To enable this, all the network nodes, namely electrical and optical (OCS) switches, as well as simulated DCN switches, are SDN-enabled and have OpenFlow 1.3+ agents installed and configured. This functional capability turns the testbed into an SDN-controlled hybrid test DCN infrastructure. The connectivity testing and experimental evaluation is

conducted using OpenDaylight (Lithium SR4 and Boron versions) SDN controller. In addition to that, the simulation framework is migrated from Windows Server OS to a Linux OS due to latency reduction and higher degree of tuning achievable. The paper also details some use-cases of the testbed with SDN control, as well as presents an outlook and strategy for testbed's performance improvement by using parallelized architecture with multi-core servers to split a large DCN simulation model into smaller subsets of DCN components.

Paper G

A Novel Algorithm for Flow-Rule Placement in SDN Switches

In this work, a novel dynamic and intelligent flow rule distribution/placement algorithm for SDN-enabled DCN switches is introduced and its performance, as well as the impact of performed flow rule migration between the expensive hardware (Ternary Content Addressable Memory (TCAM)) and "cheap" software (Random Access Memory (RAM)) flow tables on the flow processing delays is experimentally evaluated. The algorithm is implemented and executed in an ONOS SDN controller. The experimental setup is composed of a hybrid SDN switch, a set of high performance servers for traffic generation and data collection purposes, and a server hosting the SDN controller. The algorithm exploits the hybrid (hardware and software) flow table architecture of the switch with an ultimate goal of maximizing the number of Flow rules, accommodated in the switch, while limiting the potential negative effects, imposed by the flow migration process as a result of performance characteristics of the memory modules (hardware versus software). Two sets of traffic generation experiments were conducted with 150 unique data flows in each; for each experiment, 3 packet rate groups (10-20-30 and 15-30-45 packets per second (pps)) with 50 flows in each were defined, and both experiments were repeated with and without the flow rule placement algorithm enabled. The experimental results show that this algorithm allows accommodating larger number of flows, while limiting the performance degradation due to a migration process for the migrated flows and not affecting the non-migrated flows, as well as not incurring packet loss. However, stochastic latency spikes are affecting the migrated flows as a result of inherent limitations of software-based processing of the SDN switch. Additional analysis may be needed to characterize the maximum processing capabilities of both tables by loading the switch with larger number of traffic flows with higher pps rates.

Paper H

Exploring Traffic Engineering capabilities of SDN in Data Center Networks

Global cloudification of Data Center Network infrastructures, which is happening much slower than expected, but gradually becoming more and more relevant and strategically important, is gaining momentum, and there is a plethora of new cloud-based services and applications characterized by more stringent latency, jitter and bandwidth requirements. Growing operational complexity of DCNs is prompting for more efficient, optimized and easily scalable and customizable solutions. What is more, massively grow-

ing data traffic volumes are challenging the DCN operators to seek for more efficient Traffic Management (TM) solutions. SDN and Cloud computing paradigms have brought new opportunities and, at the same time, new challenges for DCN operators. In this paper, a comprehensive analysis-summary of the current best operational practices as well as the need for Traffic Engineering (TE) and TE requirements in DCNs is presented. In addition, this work provides a concise summary of the most notable TE practices, which were widely used in traditional data networks, and reasons why these approaches cannot be applied in the context of DCNs. Furthermore, the need for and benefits of an SDN-based TE approach is discussed together with open research questions, which require serious consideration and efforts of the SDN research community. The work is complemented with an analysis of TE properties and capabilities in an SDN-enabled DCN testbed in two configurations, namely a small-scale physical setup and a hybrid model to tackle the TE testing scalability aspect.

Paper I/White Paper

Evaluate Data Center Network Performance

This Technical White Paper is a result of industrial-academic collaboration between Xena Networks ApS and the Technical University of Denmark. The main goal was to emphasize the benefits and new opportunities, introduced by advanced hardware-based traffic generation and network testing solutions, such as Xena Scale (Layer 4-7) or Xena Bay (Layer 2-3), which we had an opportunity to test, for scalable, high-performance DCN-oriented research. This paper provides an overview of the main architectural properties, commonly found in traditional hierarchical DCNs with tree-like topologies as well as modern, flatter and better horizontally scalable topologies, such as Spine-Leaf structures. A DCN performance evaluation and testing study, conducted at DTU Fotonik, illustrates the research context and great application scenarios, where the functional capabilities of hardware testers can be useful. The presented case-study targeted the experimental testing of an SDN-enabled Hypercube-based flattened DCN structure. The following three aspects of scalable and comprehensive DCN testing are discussed: 1) high-speed stress-testing capabilities; 2) test scripting/automation; 3) higher resolution and accuracy of the obtained test measurement results, such as the latency and jitter.

1.5 Thesis organization

This section presents an outline of this Ph.D. thesis as well as an overview of the contents of each chapter. The structure of the thesis is visualized in Figure 1.10.

The research work, carried out as part of this Ph.D. project, consists of two parts and is composed of the research activities associated with two projects, namely "Layer 4-7 Testing at 100 Gbps" (first half of the Ph.D. project) and the EU FP7 "COSIGN" project (second half of the Ph.D. project), as well as additional research initiatives, undertaken during the Ph.D. project.

The Ph.D. thesis is structured as a research conceptualization (Part I) and a collection of associated scientific papers (Part II), co-authored/published during this Ph.D. project.

The choice of such a structure is motivated by two reasons: a willingness to provide a broader and more detailed perspective on the investigated network communication matters, and the need to link this perspective to specific research results and observations, presented in the papers.

The first part of this thesis consists of 4 (four) chapters, including this Introduction chapter, whereas the second part contains a collection of the following scientific works: one student poster, 6 conference papers, 1 journal paper as well as one Technical White Paper as a result of industrial-academic collaboration. A brief overview of chapters 1-4 is provided as follows.

Chapter 1 provides theoretical and statistical background, necessary to form a better understanding of the challenges, associated with high speed network testing and performance evaluation, detailed in Section 1.1. Section 1.2 details the concept of a network congestion and conditions which lead to this state in the context of the Internet and DCNs. Furthermore, the challenges of accurate network performance measurements, as well as the current best-practice approaches of network testing, are highlighted in Section 1.2. The content of this chapter shapes the motivation and outlines research objectives, as well as emphasizes the importance and timeliness of the conducted research work.

Chapter 2 details the TCP Congestion Control aspects in high speed Internet, focusing on the stability and performance in the context of high speed TCP connections and a wide range of associated algorithms and extensions. This chapter links to the research activities, conducted within the "Layer 4-7 testing" part of the project. The chapter is structured as a survey-style analysis, presenting the functional steps needed to establish a high-speed connection and to control data transmission (Section 2.1), as well as the evolution of traditional (Section 2.2) and high-speed (Section 2.3) and state-of-the-art Congestion Control mechanisms. Section 2.4 of this chapter is dedicated to a summarized overview of the key research findings and observations, resulting from the studies of high speed TCP CUBIC connections, focusing on the algorithmic robustness and loss recovery aspects, presented in ***Papers A and B*** and Section 2.4 itself.

Chapter 3 focuses on the performance characterization and testing aspects of DCNs, starting off with a detailed problem statement (explosive traffic growth) and the main reasons and implications of that, including the impact of new Cloud service models and Big Data, virtualization technologies and SDN/NFV deployment, as well as the Hyperscale DCs. Further, the main part of this chapter is comprised of three sections, each of which details a particular research problem investigated in this part of the project, such as:

- Designing a scalable DCN testing methodology with an overview of current standardization and best practice initiatives in DCN design, operation, and testing, as well as design and testing of a proposed hybrid DCN testbed (Section 3.1). This Section links to research activities, presented in ***Papers D, E, F, G, H and I***.
- Investigating the benefits of optical switching with WDM-based principles, applied to several DCN topologies (traditional Tree, Fat-Tree and Ring-based) at different

layers. These aspects are outlined in Section 3.2 and link to ***Paper C***.

- SDN-related testing aspects are analyzed in Section 3.3, focusing on testing of a Flow rule placement algorithm for SDN switches (related to ***Paper G***), and analysis of the benefits, challenges and opportunities of SDN-based Traffic Engineering (TE) and testing for DCNs (this discussion is linked to ***Paper I***).

Chapter 4 provides the concluding remarks, highlights open research aspects, which need additional consideration, and finalizes the Ph.D. thesis.

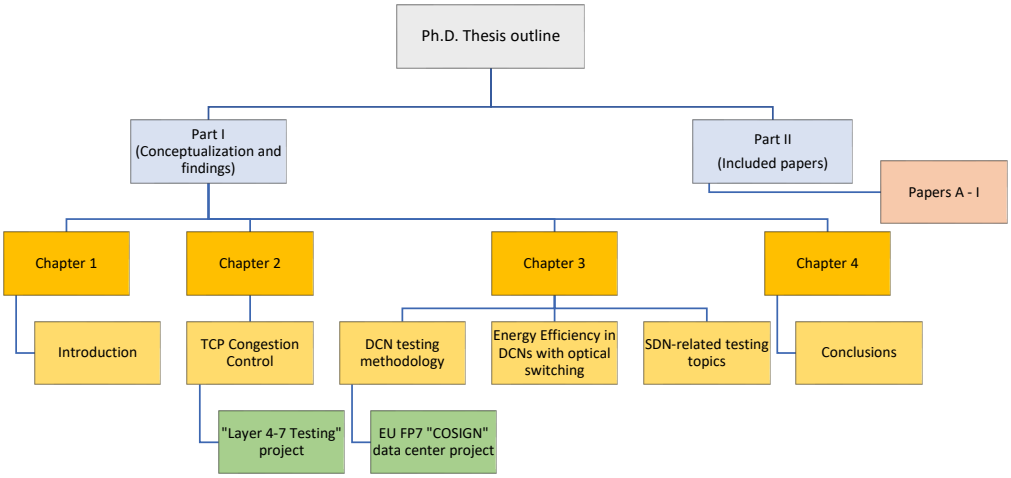


Figure 1.10: Thesis organization

CHAPTER 2

TCP Congestion Control for high speed Internet

Tremendous, nearly-exponential increase in data traffic volumes traversing the global Internet infrastructure as well as fast proliferation of new types of applications and services with very diverse Quality of Service (QoS) requirements and traffic characteristics have put a serious strain on the deployed networks and associated communication protocols, which were not initially designed for such operational scale and traffic dynamics.

If we consider Layer 4-7 of the Open Systems Interconnection (OSI) reference model, Layer 4 (Transport) is of particular importance, since this layer provides a means of logical end-to-end communication between application processes, possibly running on different hosts, over the network, and may have a significant impact on the application or service performance [76][77][78]. It is difficult to provide the exact estimates of global traffic distribution by application type or network protocols used, since the reported statistical data varies and is often limited to a subset of selected networks, where the active or passive traffic measurements were conducted, or traffic traces could be obtained. Nevertheless, available data indicates that a dominant volume of globally generated Internet traffic (60 - 90%) relies on Transmission Control Protocol (TCP)[35][79][80] as a transport protocol [81][82][83], despite some projections of increasing usage of services, which are delivered via User Datagram Protocol (UDP) [36] or UDP-based protocols [84], as well as Google's Quick UDP Internet Connection (QUIC) protocol [85][86][87](estimated 6-8% of global traffic). Therefore, the TCP protocol plays a critical role in ensuring a connection-oriented, reliable and scalable data transport service over an inherently unreliable (best-effort) IP protocol [88], and TCP's Congestion Control (CC) functionality is of paramount importance, because it provides a means of adaptive data transmission by reacting to the changes in the operational network conditions and, in this way, regulates the level of network bandwidth utilization and congestion.

Establishing, maintaining and terminating a TCP connection requires a complex interaction between multiple functional components and supporting algorithms. The lifetime of a TCP connection can be described by three main functional steps, comprised of six operational phases, such as:

1. Connection establishment

2. Data Transmission:

- ***Slow Start (SS)*** mode
- ***Congestion Avoidance (CA)*** mode
- ***Fast Retransmit (FRTX)*** mode
- ***Fast Recovery (FREC)*** mode

3. Connection Termination

The relation between these operational phases is illustrated in Fig. 2.1, where Retransmission Timeout (RTO) denotes an event, which is triggered upon expiration of a special timer, associated with outstanding data (sent, but not acknowledged yet) within a TCP connection. A transition between different operational states is triggered when a specific condition or a set of conditions is met, and since the actual conditions may differ for different TCP flavors, they are omitted for the clarity of visual presentation; thus, please refer to [89] for a general reference.

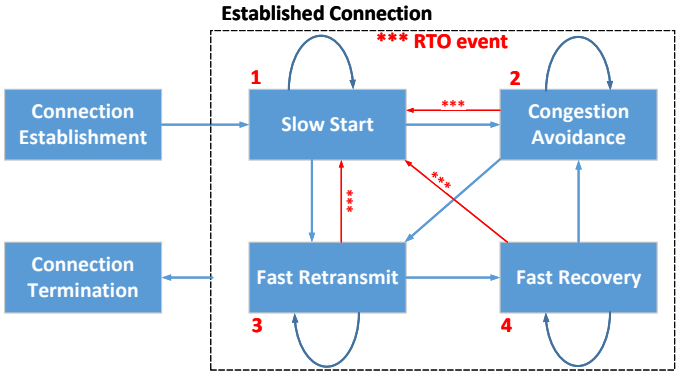


Figure 2.1: Main operational modes of a TCP connection

The main focus of this chapter is on the evolution and dynamics of an established long-lived TCP connection (see Fig. 2.1) in the context of ***Congestion Control*** and ***Packet Loss Recovery Efficiency***, encompassing different algorithms and TCP protocol extensions, which allow for more efficient data transmission in high speed data networks, including the Internet. Therefore, the actual connection establishment and termination phases will be briefly presented to form a complete view of the lifetime evolution of a TCP connection, since certain elements of these operational phases are important in the context of high speed operation of TCP, whereas the main research focus of this work was on the data transmission phase, when the connection is already established.

The outline of this chapter is as follows. In Section 2.1, three operational stages within a TCP connection's lifetime, namely connection establishment, data transmission and connection termination, are briefly discussed. The evolution of TCP congestion control

mechanisms from their early days until now is discussed in Section 2.2. In Section 2.3, we proceed with a detailed overview of a subset of high-speed TCP variants, designed for high-speed network environments, emphasizing the key pros and cons. The performance and operational stability aspects of the TCP CUBIC CC algorithm are discussed in Section 2.4 based on our research findings. Finally, Section 2.5 summarizes the chapter.

2.1 TCP connection's lifetime: operational modes

2.1.1 TCP connection establishment

Since Transmission Control Protocol (TCP) is a ***connection-oriented*** protocol [35], before any actual application data transmission can be started, a stable connection must be established first. Typically, TCP follows a client-server communication model in the connection establishment process, as depicted in Fig. 2.2 (a), where one endpoint is acting in a "client" mode by initiating a connection establishment (active Open) procedure by sending a TCP Synchronize (SYN) request to the destination endpoint, which is listening for incoming connection requests in a "server" mode (passive Open). In addition, TCP allows connection establishment in a client-client communication mode, as depicted in Fig. 2.2 (b). In this case, there is a bidirectional symmetric 2-way TCP exchange procedure, involving only one ***SYN*** and one ***Acknowledgement (ACK)*** segment, sent from both end-points. The key important steps within the connection establishment phase, valid (with specific differences noted) for both communication models, are outlined as follows.

TCP socket creation

In this step, a special data structure, called TCP Connection Block (TCB) [35], is created, which will uniquely identify every new connection to be established. Such connection descriptors are created on both endpoints, containing information, such as socket identifiers (socket number, source Internet Protocol (IP) address, source TCP port, destination IP address, destination TCP port), which create a logical binding between two end-points, and also includes other parameters, such as Initial Sequence Number (ISN) and enabled TCP extensions.

TCP state machine

TCP protocol uses a state machine to track the evolution of a TCP connection from its establishment to its termination. TCP states, relevant for the following discussion, are highlighted in Fig. 2.2 (a) and (b), depending on the connection establishment pattern used. More detailed description of various states and scope of functionality within each state can be found in the related standards [35][79].

TCP connection establishment steps

Under normal communication circumstances where a TCP connection between two endpoints is being established following a client-server communication model (Fig. 2.2 (a)), the following actions need to be performed in a 3-way communication sequence (therefore, called a "3-way handshake"):

1. A TCP client sends a TCP segment, containing ***SYN*** flag bit set, to the TCP server, announcing its willingness to set up a connection with the server.

2. When the server receives and processes client's ***SYN***, it replies with a TCP segment, which has ***SYN*** and ***ACK*** bits set. In this case, TCP ***ACK*** is confirming (assuming that ***SYN*** was not lost at this step) that the client's ***SYN*** was received, and the server indicates that it wishes to establish a connection with the client as well by adding its own ***SYN***. This message is commonly referred to as ***SYN+ACK***.

3. When the client receives a ***SYN+ACK*** from the server, it sends an ***ACK*** to confirm the reception of server's ***SYN***. At this step, client's TCP process changes this connection's state to ***ESTABLISHED***.

4. Finally, the server, upon receiving an ***ACK*** to its ***SYN*** from the client, changes the state of the connection to ***ESTABLISHED*** as well. At this point the client and the server are ready to start the data exchange procedure.

TCP Option negotiation
In addition to agreed ISNs, special TCP Options are usually negotiated during the connection establishment phase. This includes options (extensions) [90], which enable/allow high speed data transfers to be performed, such as Maximum Segment Size (MSS), Window Scaling (WS), Timestamp (TS), TCP Selective ACKnowledgement (SACK), and other extensions. This topic will be discussed in greater details in Sections 2.2 and 2.4.

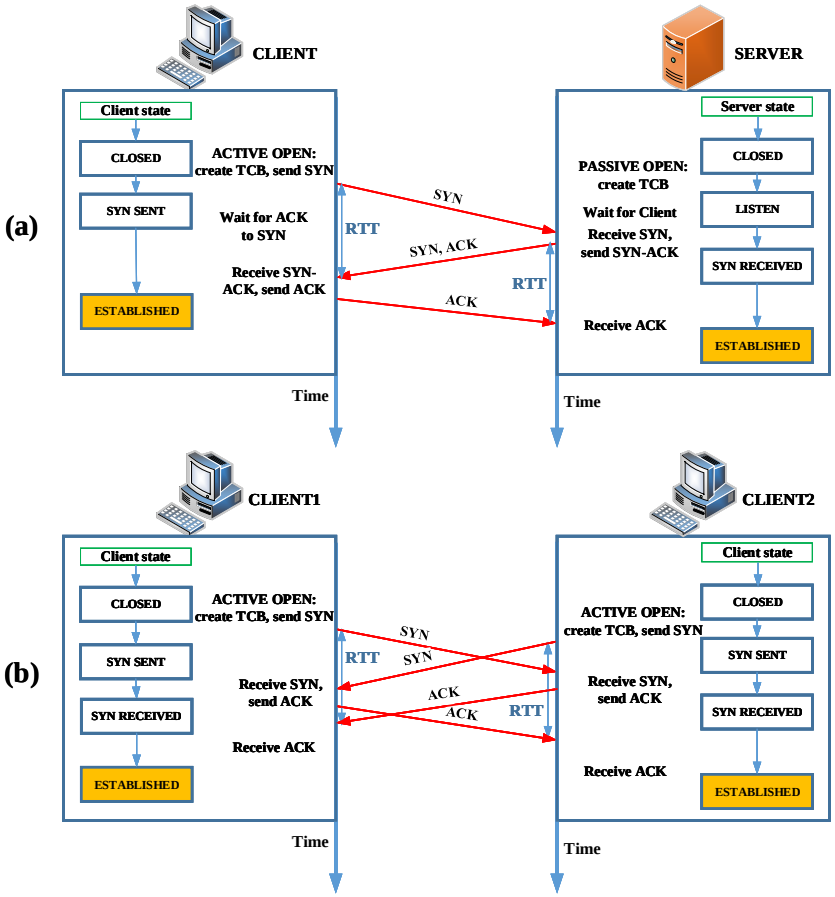


Figure 2.2: TCP connection establishment: (a) normal "Three-way handshake" procedure; (b) simultaneous OPEN procedure. Note: RTT - Round Trip Time (denotes a time period from the moment a request/data is sent until an ACK for it is received).

2.1.2 TCP data transmission

When a TCP connection is established, data exchange (unidirectional or bidirectional) may be started between the client and the server. Since the data networks have evolved to support higher transmission speeds, so did the TCP standardization efforts, defining new ways to increase the data transmission and processing efficiency. A range of TCP extensions and algorithms [79][89][91][92][93][94], beyond the general TCP connection engine [35], provide means of improving the stability and performance of TCP connections.

TCP provides a Byte stream-oriented application data transport service; however, the

data bytes are "packed" into TCP segments of a specified size (536 Bytes by default, unless a MSS option is used during the connection establishment phase, and it specifies a different value). Considering Ethernet-based [95] networks, where the Maximum Transmission Unit (MTU) size of a standard frame is 1500 Bytes (1504 for Q-tagged, or 1982 for envelope frames), the largest possible MSS value, excluding the TCP and IP protocol headers (assuming no extra protocol options carried), is 1460 Bytes.

Data transmission is governed by the Sliding Window (SW) concept [35][96], whose main aim is to ensure that the sender is transmitting the data at a rate, which is acceptable to the receiver. A set of per-connection state variables, stored in the TCB of each established connection, such as Send Window (SWND), Advertised Window (AWND) and Congestion Window (CWND), are used for this purpose. Hence, the amount of data that can be sent in one Transmission Round (TR) is determined by the SWND size, derived in the following relation [89]:

$$SWND = min(CWND, AWS) \tag{2.1}$$

In eq. 2.1, SWND size is, therefore, restricted by either the TCP receiver's advertised window size (AWS) or the dynamically adjusted CWND size, which defines the maximum amount of data (bytes or TCP segments) that is deemed safe to be injected into the network within current data transmission round before an ACK is required.

TCP-related standards [35][79] define several ways of how frequently the received data should be acknowledged: 1) one ACK per data segment received, as shown in Fig. 2.3 (a); 2) using the ***delayed ACK*** [79] mechanism, which allows offsetting (delaying) the ACK transmission based on (i) the number of data packets received (minimum every 2nd packet), or (ii) using a delay timer (maximum 500 ms, typically adjusted to much lower values for better efficiency). An example of the latter approach (delayed ACK) is depicted in Fig. 2.3 (b).

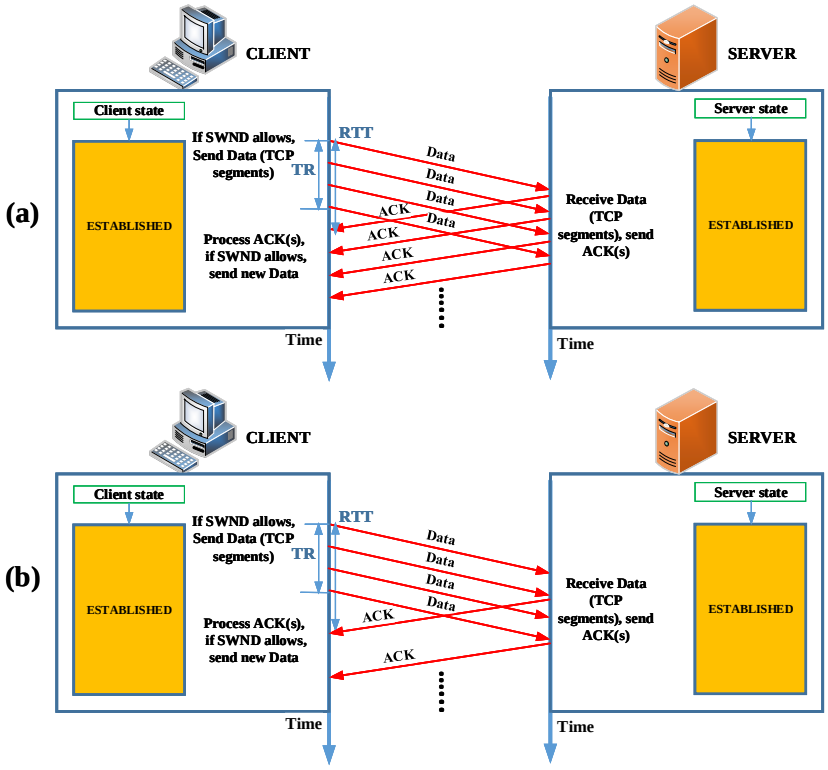


Figure 2.3: TCP data transmission: (a) acknowledging every data packet; (b) delayed ACK mechanism enabled

2.1.3 TCP connection termination

An established TCP connection can be terminated either following a regular client-server 4-way interaction model (see Fig. 2.4 (a), or a simultaneous client-server close procedure, shown in Fig. 2.4 (b). In the former, when a client-side application notifies its TCP client process that the socket resources are no longer needed (e.g., no more data to send/receive), the following steps are performed: 1) TCP client sends a TCP Finish (FIN) message (flag bit set) to the server and waits for server's ACK and FIN to arrive; 2) TCP server, upon reception of client's FIN, informs the server application to close the session; 3) When the server application is ready, TCP server sends its FIN message to the client and waits for an ACK; 4) When the TCP client receives the server's FIN message, an ACK is sent to the server; 5) Upon reception of the client's ACK, the server transitions to CLOSED state, but the client needs to wait for 2 x Maximum Segment Lifetime (MSL) time periods (120 s by default, but, in practice, may be adjustable [90]) before entering CLOSED state to make sure that the other end has closed the session and no more interaction is needed. At

this step, the reserved TCP socket resources are released and can be used (or reused) by other processes.

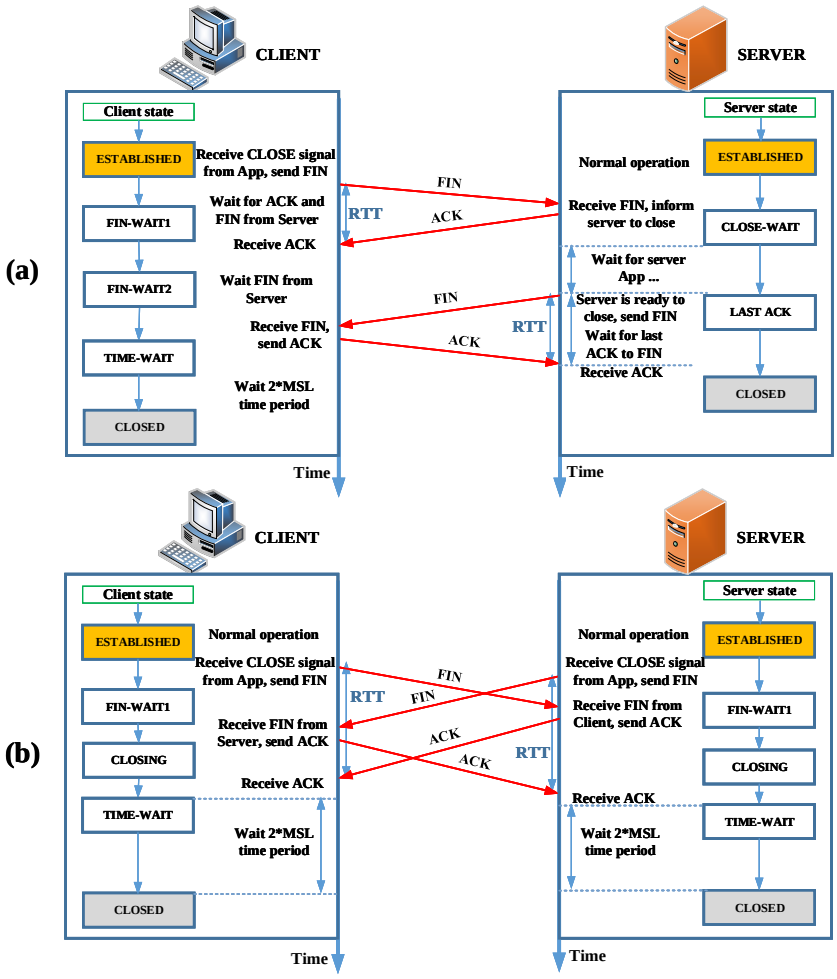


Figure 2.4: TCP connection termination: (a) regular 4-way CLOSE procedure; (b) simultaneous CLOSE procedure

In the latter case (Fig. 2.4 (b)), both end-points issue TCP FIN message to each other and wait for the corresponding ACKs; after that both sides (the client and the server, or the client-client pair) wait for 2 x MSL, before entering CLOSED state.

2.2 The evolution of TCP congestion control mechanisms

We further discuss the evolution of the Congestion Control (CC) principles, which formed the ground for modern TCP CC functionality, as summarized in Fig. 2.5.

Historically, the first documented Internet congestion collapse, which caused serious network performance degradation effects, is dated back to 1986, when the aggregate network throughput between the University of California, Berkeley (UCB), and Lawrence Berkeley National Laboratory (LBNL) dropped by three orders of magnitude, from 32 kbps to merely 40 bps [97]. The main reason that could be traced was that the CC mechanism, implemented at that time [98], was operating at the receiver's side of a Transmission Control Protocol (TCP) connection, while the bottleneck appeared to be in the network [97][99], which could not be detected due to missing receiver's bandwidth probing capabilities.

A few years later (1988), V. Jacobson proposed a redesigned TCP congestion control mechanism, which was targeting the sending side of a connection, called TCP ***Tahoe*** [99][100]. This new mechanism allowed networks to scale and to better control sharing of network resources among multiple competing flows. This was achieved by introducing a set of important algorithms, which define several operational phases (modes) of an established TCP connection, such as ***Slow Start (SS)***, ***Congestion Avoidance (CA)*** and ***Fast Retransmit (FRTX)*** [100]. By design, SS algorithm defines the first operational mode of a TCP CC framework, after a TCP connection is established and all operational parameters are negotiated and set up on both communication end-points. The data sender maintains a special Congestion Window (CWND) variable, defining the maximum number of segments that can presumably be safely injected into the network while not causing congestion at any given time. In SS mode, the sender is carefully probing the available network bandwidth by transmitting a small number of bytes (can be chosen between CWND=1 to 4 or 10 Maximum Segment Size (MSS)-sized segment(s) [101][102]) of data first, and waits for a corresponding TCP Acknowledgement (ACK). When an ACK is received, the CWND is increased by 1 MSS-worth number of bytes (or 1 TCP segment); in this case, if transmission was started from 1 TCP segment, 2 MSS-sized segments are sent in the next Transmission Round (TR), and when these TCP segments are acknowledged, the CWND is incremented to 4 segments and so on. This procedure is illustrated in Fig. 2.6.

However, following this pattern, the CWND is doubling roughly every Round Trip Time (RTT) and such an exponential increase of the amount of transmitted data might be too aggressive and would lead to a fast build up of the queues in the memory buffer of the intermediate network node(s), subsequently leading to a congestion and a loss of one or more packets when a certain queue along the path is full. In order to control the operation of SS, an additional control parameter was introduced, called Slow Start Threshold (SSTHOLD). It is used by the sender (on a per-connection basis) to manage the growth rate of the CWND. The main purpose of this parameter is as follows [103]:

1. It is an adjustable parameter, which allows the sender of a TCP connection to set an

upper limit of the CWND, which is currently deemed to be safe. Thus, this value indicates that the CWND size can be increased exponentially until this limit (an estimated boundary of a safe operation area) is reached.

2. This parameter is used as a trigger to switch the operation of the congestion control mechanism from fast exponential window growth mode (SS) to a less aggressive linear additive increase (CA phase).

In CA mode of operation, the CWND grows linearly at a much slower rate, corresponding to an increment by 1 MSS-sized segment (bytes) per RTT of the connection. In practical implementations, the size of the CWND increment is calculated by a widely used formula 2.2 [89][103]:

$$CWND = CWND + MSS^2/CWND \tag{2.2}$$

However, this method is known to be imprecise in situations with large CWND sizes, leading to more aggressive increase of the CWND size by more than 1 MSS bytes [89][104]. An alternative ***Appropriate Byte Counting (ABC)*** approach [92] was introduced and is recommended to be used to improve stability of the CA operation.

Previously, the Retransmission Timeout (RTO) event was the only indicator of a packet loss. Hence, it could take a relatively long time to recognize a packet loss when relying on a RTO event only. FRTX was introduced in ***Tahoe*** to allow the sender to detect a packet loss faster without having to wait for the RTO to occur. The packet is assumed to be lost if the sender receives ***three consecutive Duplicate ACKnowledgments (DACKs)***, containing the same Sequence Number (SN) of the next segment awaited to be received (at the receiver) [89][100][103]. The TCP connection transitions to a FRTX mode, and the determined (from the Acknowledgement Number (ACKN)) lost packet is immediately retransmitted before the RTO event, associated with it. However, one of the biggest problems with TCP ***Tahoe*** was related to the CWND update after a packet retransmission event: the SSTHOLD was set to approximately *half of the current CWND*, but the CWND itself was reset to 1 MSS-sized segment - leading to a SS process again.

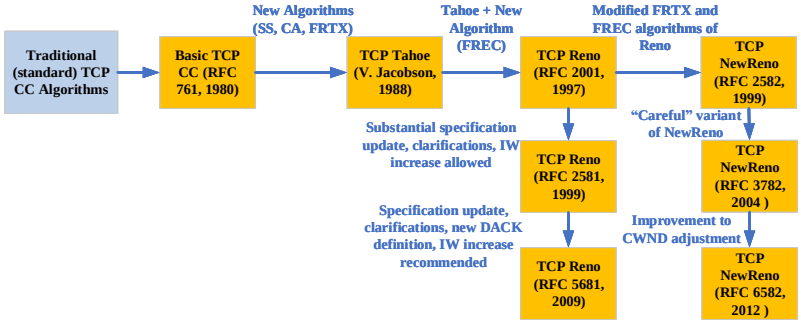


Figure 2.5: The evolution of traditional TCP Congestion Control mechanisms

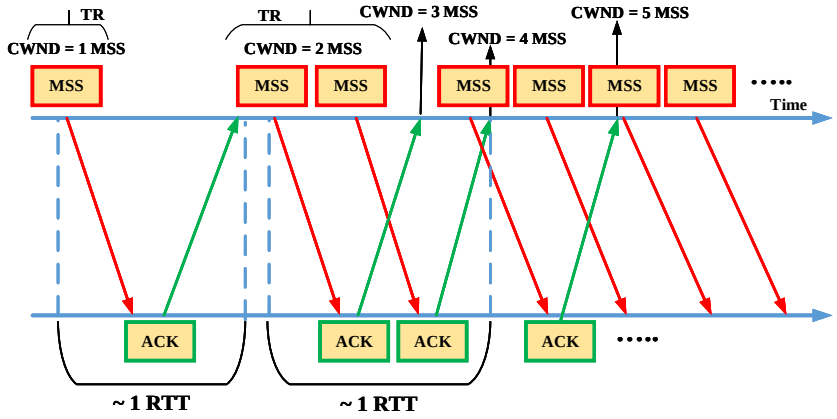


Figure 2.6: TCP Slow Start mode: exponential CWND increase every RTT (approximated)

The aforementioned problem was addressed in a TCP flavor, which is, in essence, an updated ***Tahoe***, namely by adding a new operational mode (algorithm), Fast Recovery (FREC), to the three existing algorithms of TCP ***Tahoe***, but used in combination with ***FRTX*** algorithm. Hence, ***Tahoe*** evolved into a very well-known "traditional" or standard TCP variant, known as TCP ***Reno*** [89][103]. The principle of operation in ***FREC*** mode is as follows. The main goal of this improvement is to preserve the CWND size at an acceptable (presumably safe) level after the retransmitted lost segment has been acknowledged by the receiver, avoiding CWND reduction to 1 MSS.

1. After retransmitting the presumably lost TCP segment (detected by 3 DACKs) in ***FRTX*** mode, the CWND is reduced by approximately half to CWND/2, TCP connection enters ***FREC*** mode, where it keeps track (counts) of the subsequent arriving DACKs (as a result of Out Of Order (OOO) segments, buffered at the receiver) to estimate the number of segments that might have left the network. These DACKs are used as ***ACK Clocking*** mechanism [89][103], allowing to pace out the transmission of new packets in accordance with V. Jacobson's ***packet conservation principle*** [100] (number of packets ***IN*** equals to the number of packets ***OUT***).

2. When the number of counted DACKs reaches this currently safe CWND size, TCP is allowed to send one new segment in response to a new DACK. When the lost and retransmitted segment is acknowledged, TCP exits from ***FREC*** phase and continues operation in ***CA*** mode.

However, the results of deployment and research studies of TCP Reno exposed several weaknesses in the congestion window control [89][103][105][106][107], such as poor recovery from multiple packet losses within a window and inefficient ***CWND*** adjustment

strategy after exiting from ***FREC*** phase. As a result, the TCP Reno algorithm was revised, and its updated version is called ***NewReno*** [91][105]. The main changes, introduced by TCP ***NewReno*** extension are:

1. The modification affects only the ***FRTX*** and ***FREC*** algorithms of TCP ***Reno***.

2. A new state variable, called ***"recover"***, is used at the TCP sender side of a connection to store the highest sequence number of data sent just before entering a ***FRTX*** mode, when a packet loss is suspected. Hence, when a TCP connection enters ***FREC*** mode, the packet loss recovery process will be ended only when the sender receives an ***ACK***, which covers ***recover + 1*** bytes of data (***Full ACK response***); otherwise, if the received ACK covers less than ***recover + 1*** bytes (***Partial ACK response***), the sender uses this ***Partial ACK*** to determine the next packet that is assumed to be lost, retransmits it, and returns to ***FREC*** mode.

This modification of ***Reno*** allows for more efficient recovery from multiple packet losses in a window of data (recover from ***N*** losses within ***N RTTs***), compared to ***Reno***.

These TCP extensions have been widely studied and deployed, since they provided relatively efficient congestion control and operational efficiency (i.e., resource utilization) in the network environments with low or moderate available bandwidth, or relatively short RTTs of the packet flows and very low network packet loss [108][109][110]. However, since the initial deployment of these TCP flavors, communication networks have evolved significantly, offering much larger available bandwidth and the scale of these networks, to accommodate constantly increasing traffic volumes, originating from new application and service models being deployed. These factors imposed a set of new challenges and requirements for the TCP protocol and ***CC*** functionality. As a result, extensive research studies [111][112] indicated that the well-known "traditional" TCP variants, such as ***Reno*** and ***NewReno***, suffer from severe performance degradation challenges (e.g., in terms of bandwidth utilization, loss recovery efficiency) on the network paths with large ***Bandwidth-Delay Product (BDP)***, such as in a ***Long Fat Network (LFN)*** or on satellite communication links. A ***BDP*** is defined as a product of estimated/measured bandwidth (usually limited by the bandwidth on the slowest path's link) in bits per second (bps) and latency in seconds (***RTT***), expressed as a very rough estimate of path's capacity in bits or (IP) packets. The following major reasons of such performance instability were identified [111][112]:

1. TCP ***Reno*** and ***NewReno*** are ***Additive Increase Multiplicative Decrease (AIMD)***-based algorithms with fixed increment step (1 MSS per RTT) in the Additive Increase phase, while the Multiplicative Decrease factor is 0.5.

2. TCP ***Reno*** and ***NewReno*** belong to a category of ***Loss-based CC*** algorithms, which use packet loss as an indication of a congestion (an aftermath reaction to a congestion event), leading to immediate window back-off and reduction of transmission

rate. In the high-speed context, the time it takes to recover the ***CWND*** size of a traditional TCP connection might be extremely long [111], hindering the bandwidth utilization efficiency and responsiveness to network dynamics.

3. Existing standard packet loss recovery algorithms cannot provide adequate efficiency on large BDP paths for connections with large ***CWND*** sizes. This aspect may significantly affect the ***Flow Completion Time (FCT)***.

Table 2.1: Recommended and Experimental extensions for TCP Congestion Control, Loss Recovery and detection of spurious retransmissions

Extension	Function	Enhancement	Standardization Status
RFC 7323	Extensions for High Performance	High performance operation	Standard
RFC 3168	Congestion Control	Explicit Congestion Notification (ECN) for IP	Standard
RFC 3390	Congestion Control	Increasing TCP's Initial Window size	Standard
RFC 3465	Congestion Control	TCP Congestion Control with ABC	Experimental
RFC 3540	Congestion Control	Robust ECN signaling	Experimental
RFC 3649	Congestion Control	High Speed TCP for Large Congestion Windows	Experimental
RFC 3742	Congestion Control	Limited Slow-Start for TCP with Large Congestion Windows	Experimental
RFC 5690	Congestion Control	Adding Acknowledgement Congestion Control for TCP	Informational
RFC 6928	Congestion Control	Increasing TCP's Initial Window size (enhancement to RFC 3390)	Experimental
RFC 2018	Loss Recovery	TCP Selective Acknowledgement	Standard
RFC 3042	Loss Recovery	Enhancing Loss Recovery with Limited Transmit	Standard
RFC 3517, RFC 6675	Loss Recovery	Conservative Loss Recovery Algorithm based on SACK	Standard
RFC 5827	Loss Recovery	Early Retransmit	Experimental
RFC 6937	Loss Recovery	Proportional Rate Reduction	Experimental
RFC 2883	Detection and Prevention of Spurious Retransmissions	An extension to SACK (D-SACK)	Standard
RFC 5682	Detection and Prevention of Spurious Retransmissions	Forward-RTO Recovery	Standard

A detailed analysis, presented so far, has mainly been focused on the evolution of the core per-connection ***end-to-end CC*** algorithms and their exposed limitations. However, these fundamental ***CC*** principles have been enhanced by a broad range of additional extensions, some of which were standardized and are commonly used (best-practice) in combination with these standard CC mechanisms, or are highly recommended, or

have experimental status. The last aspect means that additional tests and measurements are required to decide whether they can be safely included in a standard set of not. Table 2.1 provides a summary of the key extensions, sorted by "Function", which are known to improve the performance of TCP connections with standard ***CC*** algorithms, in the context of ***CC*** [101][113], ***Loss Recovery*** [93][106][114][115], and enable basic high speed operation [90]; in addition, experimental extensions are being tested and deployed [92][102][111][116][117][118][119][120][121]. This list of extensions is not exhaustive, but rather contains the most important identified algorithms, useful in the context of high speed TCP operation. A more comprehensive overview of a larger set of different algorithmic improvements, proposed by the research community, can be found in [122][123][124][125].

To address the first two identified reasons of performance degradation problems of ***Reno***, ***NewReno*** and their AIMD-based derivatives in high speed network environments, such as ***LFN***s with high bandwidth and possibly long delay, a range of alternative TCP variants, commonly referred to as ***"high-speed"*** TCP, have been proposed. The key changes introduced by these extensions are targeting the modification of ***CA*** phase of the ***CC*** framework discussed so far. This topic will be discussed in more detail in Section 2.3. The third highlighted problem (loss recovery) is still an open research area, but several attempts have been made to alleviate it, such as the use of TCP ***Selective ACKnowledgement (SACK)*** extension and ***SACK***-based ***Loss Recovery*** mechanisms [114][115][120], ***Limited Transmit (LT)*** enhancement, as well as a set of experimental algorithms (***Early Retransmit (ER)*** [118], ***Proportional Rate Reduction (PRR)*** [119][126]), as summarized in Table 2.1.

However, end-to-end ***CC*** mechanisms have limited knowledge of the underlying network conditions and tend to rely on different rough estimations (***BDP***, congestion events, delay variations) to adjust the operation. On the other hand, network devices can monitor the traffic load on their interfaces, and this information can be used to predict and avoid a possible real congestion by controlling the packet queue sizes at the network devices. Therefore, in practice, ***CC*** functionality is realized as a distributed combination of two sets of approaches (commonly referred to as the primal-dual approach [127]):

1. End-to-end ***CC*** by the means of TCP protocol and related algorithmic extensions (host-level ***CC***), mainly affecting the ***CC*** behavior at the traffic source.

2. Network-assisted ***CC***, realized by the means of ***Active Queue Management (AQM)*** techniques (packet queue size control), as well as using network-to-host feedback mechanism, implemented as ***Explicit Congestion Notification (ECN)*** [113], which allows network devices (e.g., routers) to inform the communication end-points of a TCP connection that the transmission rate must be reduced to avoid overloading the reporting network device and causing packet loss.

As it was stated at the beginning of this chapter (2), in this work, in relation to the "Layer 4-7 testing" project, we analyze the ***CC*** and ***Packet Loss Recovery Efficiency***

(PLRE) of long-lived high-speed TCP (CUBIC) connections, focusing on the end-to-end ***CC*** aspects. Hence, the network-assisted side of the ***CC*** framework was not investigated in this work; however, this area of CC is important, since these mechanisms allow making better-informed rate control decisions, reflecting the network congestion evolution more accurately. Thus, in order to highlight the research efforts of the industrial and academic research community and to emphasize the importance of hybrid-adaptive and high-speed ***CC*** approaches to enhance Transport layer's operational efficiency, the most well-known State-of-the-art (SOTA) ***CC*** solutions, both TCP- and non-TCP-based, are summarized in Table 2.2 and Table 2.3, correspondingly. Some of these mechanisms attracted more attention and were actually deployed in test and production network environments, as noted in Table 2.2. An earlier attempt to provide a comprehensive view on different TCP CC mechanisms can be found in a survey in [128].

Over the years of active experimentation and testing activities in the area of network transport protocols (conventionally, Layer 4 of the Open Systems Interconnection (OSI) model), and since the beginning of this research project, there have been new initiatives and fundamentally different ***CC*** solutions introduced. The research community has identified a range of problematic performance areas (aspects) within the TCP-based ***CC*** at the Transport layer, which have become critical factors, severely limiting the performance of the bandwidth-intensive and latency-sensitive applications and services in a broad spectrum of communication scenarios [129][130]. These performance aspects in the high speed context, their potential impact on the transported services, and the proposed solutions are described (as well as summarized in Table 2.2 and Table 2.3) as follows:

1. ***Low Bandwidth Utilization Efficiency.*** This factor is of particular importance for large-scale scientific applications (e.g., massive molecular or physics modeling experiments), where large volumes of generated experimental data must be distributed to the research centers for further processing, or data backup/replication activities (e.g., inter-DC data synchronization). In this case, poor scalability of the traditional ***CC/CA*** algorithms will lead to longer flow completion times, which may reduce the relevance/timeliness of the received data. To address this problem, multiple "high-speed" TCP variants have been introduced, specifically designed for high-speed large BDP networks (like LFNs), such as High Speed TCP (HSTCP) [111], Binary Increase Congestion control (BIC) [131], CUBIC [132][133], Scalable TCP (STCP) [134], Hamilton-TCP (H-TCP) [135][136], Compound TCP (CTCP) [137], Fast AQM Scalable TCP (FAST) [138], Generalized FAST (GFAST) [139], TCP Scalable Increase Adaptive Decrease (SIAD) [140], Yet Another High-speed TCP (YeAH) [141], HSTCP-LP [142], MulTCP [143], MulTCP2 [144], etc.

2. ***Poor performance in Heterogeneous networks (fixed-wireless).*** Considering that ubiquity of Internet access and consolidation of different network resources (e.g., fixed-wireless convergence) changed the way how scalable services may be provided to the end-users (reaching well beyond a single-operator network), network

Table 2.2: State-of-the-art TCP-based Congestion Avoidance and Control mechanisms

TCP CA Algorithm (flavor)	Improvements (use case)	Required modifications	Feedback (congestion signal)	TCP fairness criteria	Deployment, testing scale	Standardization efforts
HighSpeed TCP	BUE, HL, VEMC	Sender	Loss	-	Testbeds, Internet experiments, simulations	RFC 3649, E
BIC	BUE, HL	Sender	Loss	-	Large-scale, former default CC in Linux OS	-
CUBIC	**BUE, HL, C**	**Sender**	**Loss**	**-**	**Large-scale, current default CC in Linux OS and Windows 10**	**RFC 8312, I**
Scalable TCP	BUE, HL	Sender	Loss	-	Small-scale testbed	-
H-TCP	BUE, HL, C, TF, RTTF	Sender	Loss, Delay	-	Small-scale testbed and Internet experiment; simulations	IETF Draft, 2008
Compound TCP (Microsoft)	BUE, HL	Sender	Loss/Delay	Proportional	Large-scale (default: Windows Vista, Server 2008)	IETF Draft, 2008; proprietary
GFAST TCP	BUE, HL, C, TF	Sender	Delay, Loss	(α, n)-proportional	Simulations	-
TCP SIAD	BUE, HL, C, LL	Sender	Loss	-	Simulations	-
YeAH-TCP	BUE, HL, C, TF, RTTF	Sender	Loss/Delay (Queue size)	-	Small-scale testbed	-
HSTCP-LP	EBUE, HL, C	Sender	one-way Delay	-	Small-scale testbed, large-scale Internet experiments	-
TCP Vegas	LL	Sender	Delay	Proportional	Small-scale Internet measurement; simulations	-
FAST TCP	BUE, HL, C	Sender	Delay, Loss	Proportional	Limited commercial; Internet measurement, simulations	Patented, proprietary
TCP-FIT	BUE, LW, HL, VEMC, C, RTTF, TF	Sender	Loss/Delay (Queue size)	-	Large-scale Internet measurement; simulations	Patented, proprietary
MulTCP, MulTCP2	BUE, VEMC	Sender	Loss	Delay	Small-scale testbed; simulations	-
Multipath TCP	BUE, CM, HL, LW	Sender, Receiver	Loss	-	Testbeds, Internet experiments, simulations	RFC 6824, E
TCP-LP	TF, EBUE, C	Sender	Delay	-	Small-scale testbed and Internet experiment; simulations	-
TCP Westwood(+)	LW, TF, HL, HS	Sender	Loss/Delay	-	Small-scale testbed and Internet experiment; simulations	-
TCP Veno	LW, BUE	Sender	Loss/delay, network monitoring	-	Testbed, Internet measurements	-
TCP Hybla	HS, TF, RTTF	Sender, Receiver	Loss	Delay	Small-scale testbed; simulations	-
TCP-Illinois	BUE, HL, LW, TF, RTTF	Sender	Loss/Delay	-	Simulations	-
TCP BBR (Google)	BUE, HL, LW, C, V, LR, B	Sender	Congestion/network model-based	-	Large-scale testing (Google WAN), integrated in Linux kernel 4.9 and QUIC	IETF Draft, 2018

Table 2.3: State-of-the-art non-TCP-based Congestion Avoidance and Control mechanisms. ***Legend***: BUE – Bandwidth Utilization Efficiency; EBUE – Excess Bandwidth Utilization Efficiency; HL – High BDP environments, LFNs; HS – High BDP environments, Satellite links; LW – Lossy Wireless networks; C – Convergence speed; VR – Variable-Rate links; TF – TCP Fairness; WTF - Weighted TCP Fairness; B – Bufferbloat; LR – Latency Reduction; LL – Low Loss (self-induced); CM – Connection Multiplexing; SM – Stream Multiplexing; VEMC - Virtual Emulation of Multiple Connections; RTTF – RTT Fairness; TS – Throughput Stabilization; F – intra-protocol Fairness; SFCT – Short Flow Completion Time; RTC – Real-Time Communication; NLFT – Network-Level Fault Tolerance; (E) - Experimental; (I) - Informational; P - Proposed Standard; OS - Operating System. ***Note:*** this legend applies to Table 2.2 as well.

Non - TCP CA Algorithm (flavor)	Improvements (use case)	Required modifications	Feedback (congestion signal)	Fairness criteria	Deployment, testing scale	Standardization efforts
TFRC	TF, TS	Sender, Receiver	Loss/Rate	Minimum delay	Small-scale testbed, simulations	RFC 5348, P
MulTFRC	WTF, TS, VEMC	Sender, Receiver	Loss/Rate	Minimum delay	Small-scale testbed, Internet experiments, simulations	IETF Draft, 2010
CCA-RTC (Google)	BUE, TS, RTC	1. Sender(Loss+Delay); 2. Sender (Loss), Receiver (Delay)	1 - Delay, 2 - Loss	-	Large-scale testing, Internet measurement., part of WebRTC, Google Chrome, Hangouts	IETF Draft, 2015
QUIC (Google)	BUE, SM, HL, LW, C, LR	Sender, Receiver (Userspace, L5-7)	Loss/Delay	-	Large-scale testing (Google production net.), integrated with Google Chrome, YouTube clients	A set of IETF Drafts, 2015 - 2017
XCP	BUE, HL	Sender, Receiver, Router	Multi-bit signal	Max-min	Testbed, Analytical modelling, simulations	IETF Draft, 2007
VCP	BUE, HL, LL	Sender, Receiver, Router	Multi-bit signal	-	Simulations	-
MaxNet	BUE, F, C	Sender, Receiver, Router	Multi-bit signal	Max-min	Analytical modelling	-
JetMax	BUE, LW, HL, C, F	Sender, Receiver, Router	Multi-bit signal	Max-min	Small-scale testbed; simulations	-
RCP	BUE, SFCT, HL, C	Sender, Receiver, Router	Multi-bit signal	-	Small-scale testbed; simulations	-
CLAMP	VR, LW	Receiver, Router	Multi-bit signal	-	Analytical modelling, simulations	-
DCCP	BUE, LR, RTC, TS	Sender, Receiver, Router	Loss, multi-bit signal	-	Small-scale testing, Internet measurement, simulations, integrated in Linux kernel	RFC 4340, RFC 5622 (E), RFC 4342, RFC 4341
SCTP	BUE, CM, NLFT	Sender, Receiver	Loss (adapted Reno-based CC)	-	Large-scale testing, deployed for SS7 transport, ported to all major OS types	RFC 4960 and series of related RFCs

transport protocols are being challenged by the new agility/adaptability requirements. The reason is that a TCP connection may span multiple network segments with very different operational characteristics (wired (e.g., optical) links, wireless links (Wi-Fi, mobile)), so that the connection may "break" due to inappropriate rate adjustment strategy used or physical impairments of the transmission medium. To address this situation, a set of "hybrid-adaptive" TCP ***CA*** algorithms have been designed, such as TCP-FIT [145][146], TCP Westwood [147] and Westwood+ [148], TCP-Illinois [149], TCP Hybla [150] and TCP Bottleneck Bandwidth and Round-trip propagation time (BBR)[151].

3. ***Slow protocol Convergence, responsiveness.*** This aspect is important, since it defines the algorithmic speed of reaction to the fluctuating network conditions (e.g., latency variance, bandwidth availability, transient congestion/packet loss). Thus, ultimately, it affects the ***BUE***, ***FCT***, as well as may potentially lead to bursts of packet losses for connections with large CWND sizes. By design, this issue is tackled in the following TCP variants: H-TCP, GFAST TCP, TCP SIAD, YeAH-TCP, FAST TCP, TCP-FIT and TCP BBR. In certain communication scenarios fast convergence/responsiveness is also achieved by TCP ***CUBIC***, TCP Low Priority (TCP-LP) [152] and High Speed TCP for Low Priority (HSTCP-LP) [142].

4. ***Poor intra- and inter-protocol Fairness.*** This aspect defines how fair the bandwidth is shared among multiple flows of the same type (intra-protocol, RTT-Fairness) or among heterogeneous flows (inter-protocol, TCP-Friendliness), competing for the network resources on a link or an entire path. However, there is no common agreement yet on what constitutes "fairness" and fair share, and whether it is a desirable goal or not. Multiple criteria can be used to define that mathematically, e.g., as described in [153]: rate proportional fairness (the sum of proportional changes is zero or negative), max-min fairness (small flows are prioritized, smallest throughput must be as large as possible), minimum delay fairness (minimizing potential FCT), or other [153]. The choice of a criteria depends on whether we consider flows with different resource requirements (e.g., RTTs, number of links) or homogeneous flows (with the same characteristics). This aspect is claimed to be addressed by ***H-TCP***, ***YeAH-TCP***, ***TCP-FIT***, TCP ***Hybla***, TCP ***Illinois***.

5. ***Bufferbloat/latency variance.*** One of the biggest challenges faced by high-speed protocols in high speed networks is to avoid creating large packet queues in the buffers of the network devices. This problem of queue size explosion is known as "Bufferbloat", empirically described by Allman in [154], leading to increase of queueing delays throughout the Internet due to excessive traffic buffering in the networks. As a result, this leads to performance degradation of delay-sensitive flows. TCP ***CA*** algorithms, which contribute to this problem, include ***HSTCP***, TCP ***BIC***, TCP ***CUBIC***, ***STCP***. Algorithms, which attempt to keep queue size as low as

possible to address this problem are: ***Compound*** TCP, ***TCP-FIT***, TCP ***SIAD*** and TCP ***BBR***.

6. ***Bursts of self-induced packet losses.*** Flows with large ***CWND*** sizes, using high-speed TCP flavors, tend to keep the network buffer utilization as high as possible to achieve high transfer efficiency. However, since most of the high-speed flavors, including a current (de facto) default TCP CUBIC (used in Linux OS, Windows 10), are loss-based (treat detected packet loss as an indication of congestion), they tend to cause packet loss bursts and packet loss synchronization problems [112][155]. Similar observations, however, were reported in [156] for hybrid loss/delay-based ***Compound*** TCP scheme. TCP ***SIAD***, ***TCP-FIT*** and TCP ***BBR*** attempt to address this issue.

7. ***Throughput instability, fluctuations.*** This aspect is of particular importance for the Real-Time Communication (RTC) services, including live multimedia streaming over the Web. Due to specifics of media encoding (sensitivity to packet loss) and quality/bandwidth relationship, ***CC*** becomes particularly problematic. Adaptation mechanisms to account for network resource variability are implemented in TCP ***BBR***, while non-TCP-based solutions include ***Congestion Control Algorithm for Real-Time Communication (CCA-RTC)*** [157], ***TCP-Friendly Rate Control (TFRC)*** [158], ***MulTFRC*** [159] and ***Datagram Congestion Control Protocol (DCCP)*** [160].

8. ***No protection against network-level failures.*** Even though it is not part of Tranaport layer functionality, this aspect is useful when a critical uninterrupted communication session is required - if one TCP connection within a session fails due to network reachability problems on one of the interfaces, a connection on another available network interface may be used in parallel and protocol logic would switch-over to an active path. Another application area is Layer 4 traffic load balancing over multiple available interfaces. Such capabilities are supported by ***Multipath TCP*** [161].

Other, non-TCP-based network-assisted mechanisms, as it can be seen in Table 2.3, designed for high-speed operation, include ***Explicit Control Protocol (XCP)*** [162], ***Rate Control Protocol (RCP)*** [163], ***Variable-structure Congestion control Protocol (VCP)*** [164]. ***MaxNet*** [165] targets improvement of convergence, ***BUE*** and intra-protocol fairness, while ***JetMax*** [166] is reported to improve these aspects in heterogeneous high-speed networks. These alternative CC protocols are claimed to be more robust and adaptive to changes in the network due to more efficient feedback (control loop) mechanism, but the main problem with these solutions is that almost all of them require substantial modifications of both sender and receiver, as well as support of ***ECN*** and related extensions in the routers for multi-bit congestion signalling. Some well-known UDP-based solutions, such as Google ***Quick UDP Internet Connection (QUIC)*** [85]

protocol suite, provides ***CC***, ***Loss Recovery*** and ***Security*** functions in the user-space (Layer 5-7).

In the next section, the main characteristics of the high speed ***CC*** algorithms are discussed, emphasizing the algorithmic differences and pursued design goals.

2.3 High Speed TCP congestion control algorithms

In this section we will discuss the operational aspects of a set of advanced Transmission Control Protocol (TCP) Congestion Control (CC) techniques, which were designed to address the limitations and performance degradation problems of traditionally deployed algorithms, such as TCP Reno and NewReno, as it was detailed in Section 2.2.

Some well-known high-speed TCP flavors, such as High Speed TCP (HSTCP) [111], Binary Increase Congestion control (BIC) [131], CUBIC [132][133] and Compound TCP (CTCP) [137], have been studied analytically, via simulations, as well as in experimental and production deployments, and are reported to offer a significant improvement in terms of higher throughput, bandwidth utilization efficiency, and more effective response to network dynamics (e.g., a congestion event, delay variations or increase of the available bandwidth) [111][132][137]. However, there is a range of scientific works, arguing that many modern high-speed TCP flavors tend to be much more aggressive in terms of bandwidth sharing fairness (get much higher bandwidth than their fair share [112][130][167][168][169]) among multiple different TCP connections, potentially sharing the same network path or link. In addition to that, when network resources have to be shared among loss- and delay-based (or hybrid with a delay-based component) TCP flavors, such as ***CUBIC*** and ***Compound***, loss-based flows take over most of the available bandwidth, severely degrading the performance of delay-based flows [168]. Another noticed problem, as reported in [155], occurring in large BDP networks with multiple TCP CUBIC connections, is loss synchronization among multiple connections, leading to CWND reductions and throughput drop over time. Newer solutions were introduced to address some of the limitations of the aforementioned high-speed TCP flavors, focusing on the operational efficiency in heterogeneous networks, faster protocol convergence, better TCP-friendliness, delay reduction, and other factors, as outlined in Section 2.2 and Table 2.2. However, as SOTA literature analysis showed [122][123][124][125][140][151], the majority of the proposed solutions perform well in specific communication scenarios, with limited experimental testing and validation. Standardization process of network transport protocols is another important area of discussion, since it is known to take a ***very*** long time from the initial idea/proposal to the actual large scale (Internet-wide) deployment, causing serious debate and leaving many open research questions.

In order to better understand the reasons and implications of such behavior of different high speed flavors, it is important to expose their algorithmic properties. Since there is already a wide range of algorithms designed for high speed communication purposes, we will limit the scope of our discussion to a subset of algorithms, which are worth particular attention and represent the main traits of their respective category. High speed operation of TCP flavors is made possible by combining several essential components, such as:

1. ***Window Scaling (WS)*** TCP option supported and enabled on both end-points. Negotiation of the supported TCP options is performed at the connection establishment phase, as it was explained in Sub-section 2.1.1.

2. A ***CA*** algorithm with good scalability and stability properties. These algorithms mainly affect the TCP connection's operation in the ***CA*** phase (recall Fig. 2.1 and discussion on page 29 and 30). Hence, the name ***CA algorithm*** does not necessarily imply the CA properties of an algorithm, but rather relation to a CA phase of a TCP connection.

3. Additional supporting extensions (see Table 2.1) for improved Loss Recovery, detection/prevention of spurious retransmissions and more accurate RTT estimation, helping to sustain high throughput and ensure faster reaction to variations in the network state.

Depending on the source of ***Congestion information*** used to detect or predict and avoid network congestion, the existing high-speed TCP ***CA*** mechanisms can be grouped into four main categories, as shown in Fig. 2.7 and summarized in Table 2.2.

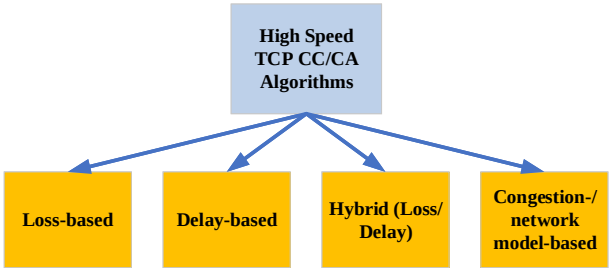


Figure 2.7: High Speed TCP CC/CA algorithms: grouping by the congestion signal type

We further present a comprehensive view on the reasons for having multiple different categories of ***CA*** algorithms, as well as existing operational limitations of algorithms from different categories.

1. ***Loss-based CA mechanisms***. This category of CA algorithms uses packet loss (detected by the means of 3 duplicate ACKs, or additional methods) as an indication of occurred network congestion - the same principle as in TCP Reno and NewReno. Examples of such algorithms include, among others, TCP ***CUBIC*** and ***HSTCP***. While possessing good scalability characteristics to achieve high bandwidth utilization in large bandwidth and long delay networks by applying non-linear ***CWND*** increase and flexible decrease (other than 0.5 of CWND) strategies, this algorithmic non-linearity may actually create extra performance degradation problems. If we consider a situation when multiple high speed flows with large ***CWND*** sizes share

the resources of a common bottleneck link (or a set of links), a buffer overflow situation may occur, leading to large bursts of packet losses, as studies indicate [155][170]. Loss-based high-speed algorithms tend to keep the network buffer utilization at the maximum level to achieve high efficiency [167]. As a result, long-tailed (standing) queues may build up in the network buffers, increasing the average packet queueing delays, potentially leading to a "bufferbloat" effect, introduced earlier. The problem with this approach is threefold. First, as a literature review shows [151], packet loss is not necessarily equivalent to congestion: packet loss may occur due to transient traffic bursts in networks with shallow buffers, even when the link is mostly idle. In such a situation, reactive window reductions hinder the possibilities to achieve high sustained goodput ("useful" data rate, i.e., not including any retransmissions) due to the requirements of low packet loss (e.g., to sustain 10 Gbps on a link with 100 ms RTT, packet loss below 0.000003% is needed). Second, loss-based algorithms, such as CUBIC and HSTCP, cannot efficiently distinguish a real packet loss from packet reordering or transient delay spikes, since they modify only ***CA*** phase of a TCP connection, and still rely on the baseline algorithms of Reno/NewReno for this purpose, which are known to malfunction in a range of scenarios [91][171][172][173]. The third problem with using loss-based-only CC algorithms for high speed connections with large CWND sizes is related to the operation in the ***SS*** phase where the available network bandwidth is "probed" by exponentially opening the CWND size: depending on the configured ***SSTHOLD*** value, too large burst of packets may be sent out, overshooting the network buffer's capacity. This is the case in the situations, when there is a sudden decrease in the available bandwidth [174], and as a result of relatively slow convergence of these algorithms. It has been detected that packet loss bursts are very common at this step (Slow Start) [170], leading to prolonged recovery and flow completion times. There have been several attempts [170][175][176] to limit the severity of potential performance degradation by modifying the ***SS*** operation to allow adaptive reduction of the CWND increase rate when approaching the ***SSTHOLD*** area, where switching to the ***CA*** phase is performed.

2. ***Delay-based CA mechanisms***. An alternative approach was introduced, taking the delay component into the transmission rate adjustment "equation". Probably, one of the most historically well-known (not the first though) delay-based algorithms in this category is TCP ***Vegas*** [175]. We will briefly present its principles of operation, since several high-speed TCP algorithms were developed based on Vegas. In this case, the transmission rate (hence, the ***CWND*** size) is linearly adjusted based on the comparison result (the difference) of two tracked rate variables, namely the estimated throughput and the actual throughput, where the throughput estimation is based on RTT measurements. More specifically, the estimated throughput is calculated once per RTT as a ratio of ***CWND/baseRTT***, where ***CWND*** is a current congestion window size and baseRTT is a minimum of all RTTs, measured during

the connection's lifetime. The $Actual$ throughput is computed as ***BytesSent/RTT***, where RTT is a current RTT sample and $BytesSent$ specifies the number of bytes transmitted within a RTT tracking period for a chosen packet. The rate adjustment procedure is performed once per RTT in the following manner:

$$\boldsymbol{Diff} = Estimated - Actual, \tag{2.3a}$$

$$\textit{if } \boldsymbol{Diff} = \alpha, \textit{ increase CWND} \tag{2.3b}$$

$$\textit{if } \boldsymbol{Diff} > \beta, \textit{ decrease CWND} \tag{2.3c}$$

$$\textit{if } \alpha < \boldsymbol{Diff} < \beta, \textit{ no changes} \tag{2.3d}$$

Where α is a threshold of lower bound of the amount of extra data that can be sent to the network, and β is an upper bound threshold of the maximum amount of extra data allowed. Both thresholds are expressed in kilobyte per second (kB/s) or using network buffer (number of router "buffers") notation [175]. As a result, Vegas is trying to maintain the actual transmission rate within the $\alpha - \beta$ range, not self-inducing packet loss, while being able to utilize the extra available bandwidth. The key important aspect here is that the algorithm does not use the packet loss, but rather the increase of the delay as a congestion signal. It "senses" a potential future congestion by using timestamp-based RTT measurement (more accurate), and avoids causing a congestion by backing off the CWND size (hence, the transmission rate) before the network queues are filled up by this connection. Hence, this delay-based algorithm belongs to a ***CA*** category of ***CC*** mechanisms. TCP Vegas still supports the ***FRTX/FREC*** loss recovery of Reno, but adds modifications allowing to retransmit presumably lost packets earlier (e.g., after the first or second DACK), as well as supports modifications of the ***SS*** phase to limit the CWND increase rate (allow exponential window increase by 1 MSS only every other RTT). Even though this algorithm was an important step forward in better understanding and designing new delay-based ***CC*** principles, it was never deployed in the Internet due to a serious limitation: it has been shown in multiple studies that Vegas cannot compete (in terms of bandwidth utilization efficiency) with the loss-based algorithms in a shared network environment [177][178]. The problem is that the "congestion avoidance" step is activated before a congestion occurs by reducing the CWND, while loss-based algorithms (e.g, Reno, NewReno, CUBIC, etc.) continuously increment their windows until the buffer is overfilled. Also, it is reported that performance of Vegas starts degrading when the RTT of the path exceeds 50 ms [179]. Therefore, Vegas is able to ensure fair bandwidth allocation only when all the competing flows use Vegas algorithm. Several modifications of Vegas were proposed to remedy some of the detected problems, such as TCP ***Nice*** [180], ***CODE*** TCP [181], ***NewVegas*** [182] and ***Fast AQM Scalable TCP (FAST)*** TCP [138].

We will focus on the ***FAST*** TCP in the context of our discussion, since it is a variant, specifically designed for high-speed network environments. It pursues two main

goals: 1) it retains the steady-state properties of Vegas, namely it does not penalize performance of connections with a large RTT and, in this way, achieves weighted proportional fairness; 2) it introduces a more responsive non-linear CWND control function to improve network resource utilization efficiency. As compared to Vegas, in FAST TCP, the ***CC*** mechanism is split into four functionally independent components, which can be maintained separately, such as: ***data control*** (which packets to send), ***window control*** (how many packets to send), ***burstiness control*** (when to send these packets). The information for these decisions is provided by the ***estimation component***. What is more, FAST TCP uses queue size estimation α, which denotes the number of packets queued in a router along the flow's path, and scaling factor $\gamma \in (0, 1]$, in addition to the estimated throughput (recall $CWND/baseRTT$ in Vegas). Despite all the introduced improvements, FAST TCP algorithm is reported to have several performance problems, such as: 1) performance metrics (throughput, packet loss rate, link utilization) are highly sensitive to α parameter setting [183][184]; 2) inter-flow fairness (of FAST TCP flows to flows with differing RTTs) may be degraded due to a ***persistent congestion problem*** [112][183]. However, some works argue, that good protocol stability, fairness and bandwidth utilization may be achieved by proper adjustment [185] or dynamic tuning (algorithm's modification) [139][186] of FAST TCP. Nevertheless, one of the key factors, hindering the deployment possibilities of delay-based algorithms is the ***accuracy of delay-based congestion predictors***. Multiple studies concluded that ***delays (RTTs) are, in general, weakly correlated with congestion*** [187][188][189]. This is influenced by the following [124]:

- Bottleneck buffer size (determines the algorithmic aggressiveness of delay-based variants) may have significant affect on distinguishing a real delay variation from a measurement noise;
- RTT measurement issues (due to too coarse-grained timers, RTT undersampling, TCP segmentation offload introduced measurement noise, use of Delayed ACK) [190];
- RTT sampling and level of statistical multiplexing (the number of flows) [188];
- The impact of wireless links (due to link-layer scheduling, error recovery, etc.) [191].

3. ***Hybrid Loss/Delay-based CA mechanisms***. As new applications and services become more feature-rich and resource-intensive, the underlying networks grow in scale and complexity, and it has become apparent that relying only on packet loss or only a delay variation signal to define the evolution of high speed TCP connections may be inefficient in modern high speed, especially heterogeneous, networks. A relatively new category of hybrid algorithms, combining the benefits of both previously discussed groups, has emerged and attracted active research and

development interest, as it was first introduced in Section 2.2 and Table 2.2. We will limit the scope of our brief discussion to ***CTCP*** [137], since it well represents the hybrid group.

Compound TCP [137][192] is an algorithm, developed by Microsoft, in an attempt to achieve the following properties: 1) maintain good TCP-friendliness to standard TCP flows; 2) improve bandwidth utilization on long-distance high-speed Internet links; 3) proactively avoid causing packet loss due to the buffer overflows. This was achieved by combining a standard TCP Reno (loss-based part) with a delay-based component (derived from TCP Vegas), which is defining the CWND increase/decrease rate. The delay-based module uses the $Estimated$ and $Actual$ throughput (as in Vegas) and multiplies the difference value by the $baseRTT$ to obtain an estimation of the ***number of backlogged packets*** (buffered in the router), instead of just a throughput difference. Hence, the TCP connection maintains two separate Windows, the CWND (which is a loss-based window) and the ***Delay Window (DWND)***, and the DWND component complements the CWND only in the CA phase of the connection (otherwise, it is disabled in SS and upon a detected packet loss) to achieve scalability and CA. A set of key tunable parameters define the operational properties of CTCP, such as γ, which denotes a threshold of the number of backlogged packets to decide whether the DWND should be increased or decreased, α (scalability), β (reduction smoothness), k (responsiveness) and ξ, which defines how fast the DWND component should reduce this window when an early congestion is detected. The importance of this algorithm is in the fact that it was actually deployed by Microsoft in their production network, and included as a default CC mechanism since Windows Vista and Windows Server 2008 OS versions. However, more detailed performance studies revealed some serious limitations, such as:

- Dependency on the network buffer size (instability, large queue size variance with large network buffers deployed) [156];
- Sensitivity to the queueing delay variations and packet loss leading to fluctuations in output rate [193];
- Sensitivity to the queue management approach used in the routers, e.g., poor performance and instability of the queue size in the routers with large buffer regime and Random Exponential Marking (REM) AQM discipline [156];
- Remaining bandwidth utilization problems in shared environments with loss-based algorithms (e.g, CUBIC) [168][169].

Some proposed solutions to reduce the impact of these problems include: joint design of the network buffers (sizing, AQM discipline parameter tuning) with Compound parameter tuning [156], or random packet dropping and virtual connection concatenation (using proxy servers) [168]. However, most of these mechanisms are intrusive - require modifications of either the parameters of the network devices or

the use of extra network devices, and, therefore, is practically impossible to achieve in large-scale Internet deployment scenarios. Finally, as of ***Windows 10 OS*** (e.g., ver. 10.0.16.299.x) the default CC algorithm used is ***TCP CUBIC*** (ver. 11) .

4. ***Congestion-/Network model-based CA mechanisms***. This is a new category of CC mechanisms, and a first prototype of such a mechanism, called TCP ***Bottleneck Bandwidth and Round-trip propagation time (BBR)*** [151][194] was developed and offered for evaluation by Google in 2016. Since then, this TCP extension was actively tested internally and deployed in Google's production network environments (e.g., B4 WAN, being deployed for YouTube, Google.com). The main idea behind this extension is that in order to achieve operational stability and high bandwidth usage efficiency, two conditions must be satisfied: rate balance (the bottleneck packet arrival rate equals the bottleneck bandwidth) and full pipe (the total data in flight equals to the BDP). When these conditions are met, this allows achieving two goals, namely to reach the highest possible resource utilization efficiency (up to 100%) and to prevent bottleneck starvation by guaranteeing that there is enough data to send without overfilling the pipe. Instead of relying on the packet loss and queueing delay variation as two isolated components of the control loop, BBR is using two path constraints, such as ***Round-Trip propagation delay (RTprop)*** and ***Bottleneck Bandwidth (BBW)*** to build a logical "network path model" to derive ***congestion state estimates***. Thus, BBR consists of two core functional components, such as: 1) Processing on ACK arrival and 2) Data sending. When a new ACK arrives, the former is responsible for updating the ***RTprop*** estimate and recording the amount of delivered data from the moment a distinguished data packet was sent (for which the RTT is being measured) until an ACK for it was received. The latter is used to pace (match the packet arrival rate to the bottleneck link's departure rate) every packet transmission by using a ***pacing rate*** control parameter, and a ***cwnd gain*** parameter sets an upper bound on the in-flight data to a small multiple of the path's BDP in order to account for different receiver-side communication specifics (Delayed ACKs, Stretched ACK generation, ACK aggregation, etc.). The most important difference between the CC mechanisms discussed before (in this section) and TCP BBR is that BBR maintains a dynamically adjustable network path model, where previously mentioned path constraints (***RTprop*** and ***BBW***) are being continuously updated from the measurements by applying a set of filters (functions) to reflect the latest detected changes in the ***RTprop*** and ***BBW***). Therefore, the transmission rate and amount of data to send is determined by these two parameters, since BBR tries to maintain a BDP-worth ($BDP = RTprop \times BBW$) amount of in-flight data in order to reduce the possibility of queue buildup and latency increase, and filters control the adaptation of these parameter values. For example, for every received non-DACK, BBR collects ***BBW*** estimation samples in a measurement window (bucket), typically chosen as 6-10 RTTs, and a maximum BBW rule (choosing the largest recorded sample value) is used to update this estimate. As for the RTprop

measurements, the same windowed filtering approach is used, where a typical measurement interval is chosen between 10s of seconds to several minutes, and the RTprop value is filtered using minimum RTT rule (choosing the lowest recorded value within this window).

The following set of performance evaluation results, depicted in Fig. 2.8 and Fig. 2.9, illustrate the performance improvement of a BBR-based flow as compared to TCP CUBIC . These results provide an important insight to the key algorithmic differences between these two algorithms, which are currently de facto default CC algorithms, deployed on a large near-Internet-wide scale. Fig. 2.8(a) shows how the amount of delivered data evolves at the very beginning of the connection (the first second) for a BBR flow and a CUBIC flow under the same operational conditions. The BBR algorithm handles all the events (ACKs, when to send, etc.) using a notation of "states" (***Startup, Drain, Probe BW***), which are associated/mapped to a table with fixed gains and exit conditions (when to move from one state to the other). The ***Startup*** and ***Drain*** states are used at the connection's startup, followed by the ***probe BW***, in the following way (see Fig. 2.8(a)):

a) In ***Startup***, BBR tries to use binary search for the ***BBW*** by using a gain factor of 2/ln2 to double the sending rate (within a measurement window); in this way, the ***BBW*** is discovered in approximately $log_2 BDP$ RTTs, but such an exponential rate increase creates an excessive queue of up to $2 \times BDP$, when the maximum in-flight is clamped to $3 \times BDP$ (Fig. 2.8(b)).

b) Then, when the ***BBW*** is detected, BBR transitions to ***Drain*** "state", where an inverse of the Startup gain (i.e., ln2/2) is used to reduce the rate and drain the excess queue size. Finally, the BBR algorithm switches to ***probe BW*** phase to continue cyclic probing of the BBW in the steady state.

The key observation here is that BBR, after a fast boost of the initial sending rate (to detect the BBW) starts immediately reducing the sending rate to eliminate the created standing queue and equalizing the rate to match the BBW, whereas CUBIC cannot do that without network path estimation capabilities (to determine how much of in-flight is excessive) and continues loading the path and creating a queue bloat. This situation is depicted in Fig. 2.8 (b) in a form of RTT evolution over time, limited to a 1-second observation time span for the presentation clarity: CUBIC initially probes the bandwidth in less aggressive steps than BBR, but continues to increase the window until the buffer is overfilled and loss occurs, and that ultimately leads to a linear latency increase. Additional observation is provided by Fig. 2.9, showing how the RTTs of both connections evolve over the investigated 8-second time span. TCP CUBIC quickly fills the available buffer space, reaches overflow state and then cyclically continues filling the buffer from 70% to 100% (due to new data and loss recovery retransmissions), while the BBR flow runs with a negligible queue size and low latency (at the RTprop level), after the startup step

at full bandwidth. This is the main property of BBR: elimination of the standing queues to minimize the latency, while quickly adjusting the new sending rate to the instantaneous network conditions (path properties).

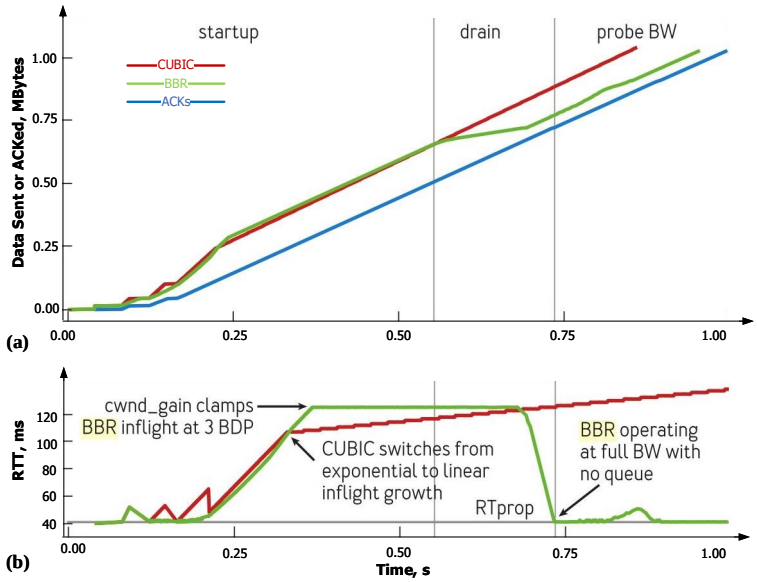


Figure 2.8: The first second (startup) of a 10-Mbps 40-ms flow evolution: comparison of BBR and CUBIC [194]. (a) Delivered data; (b) RTT evolution

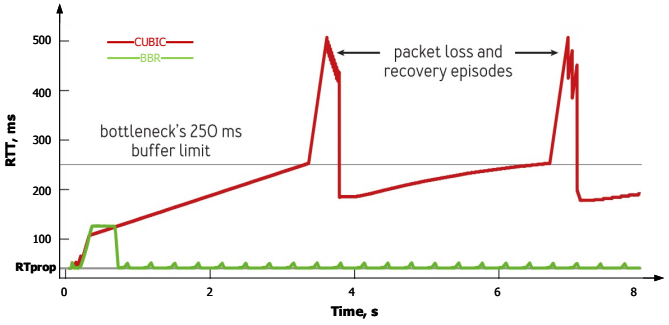


Figure 2.9: The first 8 seconds of a 10-Mbps 40-ms flow evolution: comparison of BBR and CUBIC [194]

A wide range of important TCP-based high-speed communication aspects was discussed so far, emphasizing the operational properties of and differences between various CC/CA extensions of TCP. As practical deployments and research studies show, there is no universally applicable solution available, which can satisfy all the existing constraints, as well as performance (throughput, latency, queue size control) and stability (fairness, fast convergence) requirements. The available research results are often incomplete, difficult to reproduce or even contradicting, since there is a wide range of measurement and testing assumptions introduced, and their overall impact on the accuracy of the presented results is often not taken into consideration. The performance studies of TCP BBR and its coexistence with TCP CUBIC, conducted by Google [194], indicates that TCP CUBIC, while being the most widely deployed CC algorithm on the Internet, can greatly contribute to the congestion in high speed networks, causing packet loss bursts, prolonged loss recovery times and long-tailed queues (bufferbloat). TCP BBR seems to be a promising mechanism, but it's a completely new solution, which still requires extensive testing and parameter tuning. Therefore, it is obvious that there is a wide range of open research issues in the area of high speed network congestion control, which need to be addressed. This note summarizes our discussion of high-speed TCP congestion control mechanisms and sets the context for the next section, dedicated to the performance and stability analysis of high-speed TCP CUBIC connections, based on our research findings as a part of this Ph.D. research project.

2.4 TCP CUBIC: performance and open research issues

The focus of this part of the conducted research project in the context of Layer 4-7 testing was on the robustness (operational stability) and loss recovery efficiency of high speed TCP CUBIC connections. This CC algorithm was chosen for evaluation, because at the beginning of the Ph.D. research project, this TCP flavor already was a default CC module of Linux OS, which is used on an Internet-wide scale as a platform to power a large number of Internet servers, providing a wide range of feature-rich services. According to the latest statistical data, provided by the W3Techs resource [195], "Unix is used by 67.1% of all the websites whose operating system we know", and "Linux is used by 54.5% of all the websites who use Unix" [195], as well as the Linux foundation ("Linux is the operating system for over 95% of the top one million domains" [196]). This is a clear indication of massive proliferation of this OS type; hence, the decisions made by the members of the Linux Foundation [196] regarding the choice of the CC mechanism, positioned as "default", have significant consequences on the operational stability of the Internet. Therefore, as network and service deployment grows, so does the operational complexity and performance requirements for the network protocols, where the Transport layer (Layer 4) plays one of the most critical roles in ensuring smooth networking experience. In addition, as it was mentioned in Section 2.3 (p. 54), the newest Windows 10 OS is shipped with TCP CUBIC as a default CC algorithm used. We further present a summary of the methodology used and the key findings, resulting from our research activities, while a more detailed analysis and the results can be found in the related papers.

Robustness of high speed CUBIC connections to severe operating conditions

Since its initial deployment, the TCP CUBIC CC extension, designed for fast long-distance networks, has been studied analytically, in simulation and experimental testbed environments. It has been shown that this algorithm greatly improves the resource utilization efficiency and offers good inter-flow fairness and convergence speed in specific communication settings, but can be too "aggressive" and continuously lead to self-induced packet loss bursts, loss synchronization patterns, explosion of the queueing delay and long loss recovery times for flows with large CWND sizes on large BDP paths (as detailed in Section 2.3 on p. 48 and p. 49).

One of the biggest challenges in network protocol testing is the fact that an algorithm may be implemented in different ways in the TCP/IP stacks of different OSs, sometimes not conforming to the IETF specifications, because specific changes may need to be made to incorporate new congestion control elements into the kernel space, and that may not always lead to expected performance. There have been several attempts to expose such issues in the Linux OS kernel [197][198][199][200]. The goal of this simulation-based study, outlined in ***Paper A/Poster***, was to perform conformance tests of the algorithmic properties of the implementation of TCP CUBIC in a simulation tool (Riverbed Modeller) and assess its (TCP CUBIC's) performance characteristics. Additional algorithms to stabilize the CWND adjustment of Reno/NewReno (***Appropriate Byte Counting (ABC)***) and fixes to ***SSTHOLD*** and CUBIC's CWND adjustment were implemented and validated in [104]. This work (***Paper A/Poster***) was further extended in ***Paper B*** with an aim to complement the existing results with additional insight on how highly difficult operating conditions (severe packet loss under increasing RRTs) affect the operational robustness and adaptation capabilities of long-lived high speed TCP CUBIC connections, operating with large CWND sizes in high-bandwidth and long-delay networks (e.g., Long Fat Network (LFN)-like environment).

Paper A/Poster sets a preliminary roadmap of high-speed TCP-related performance evaluation activities, which were planned at the beginning of the first part (Layer 4-7) of this project. We considered the impact of two packet loss patterns, namely a random and a synchronized (network buffer overflow) pattern, which were further explored in ***Paper B*** and work on Packet Loss Recovery Efficiency (paper under preparation), respectively, in more details. The main findings and observations, which can be derived from the conformance and performance testing results for TCP CUBIC, covered in ***Paper A/Poster*** and ***Paper B***, are as follows:

1. The TCP CUBIC algorithm is susceptible to both loss patterns, namely random and synchronous, and even with additional introduced tweaks (e.g., setting a maximum burst limit size per transmission round, adjusting the value of the scaling factor C), the maximum obtained bandwidth share of CUBIC flows is decreasing with the increase of RTT (even though the CWND growth is dependent only on the loss-free time period) and throughput variance (instability) is the largest in the high loss region ($BER = 10^{-5} - 10^{-7}$) under the random loss pattern. As

regards synchronous loss tests (conformance tests) with buffer overflow scenario, the validated implementation of CUBIC in the simulation interacts with the network buffers as expected and reported in other studies: increasing number of high-speed CUBIC flows create a cyclic buffer overflow pattern, leading to queue size explosion and latency build-up. This behavior was confirmed in the latest studies of Google, as it was discussed in Section 2.3 (p. 55), where TCP BBR was tested together with the latest CUBIC (ver. 11), including the hybrid slow start [170] mechanism enabled.

2. When dealing with large-scale Layer 4-7 testing, it is important to consider the scalability aspect of the testing platform itself. The architecture and configuration parameters of a TCP connection engine can be optimized and limited to a basic set of supported functionality, for example, by adapting certain light-weight TCP/IP stacks, such as listed in [201][202] (e.g., light-weight-IP, uIP) widely used in resource-constrained embedded systems (e.g., IoT devices). However, while these compact stacks are efficient enough in a distributed (per-device) fashion, where each small system needs only a basic TCP/IP connectivity for some data collection and exchange, they may not be easily integrated into large-scale testing solutions, due to the fundamental architectural limitations: they heavily rely on kernel space system calls (overhead), interrupt-based event processing (call-back functions), and offer poor scalability on multi-core systems. Since support of multi-core architectures on high performance testing platforms is one of the key factors, defining the scalability properties of such systems, this is one of the main criteria when choosing a TCP stack to integrate. Hence, new TCP/IP stacks, designed to scale well on multi-core systems, and built around the user-space, rather than the kernel-space, such as mTCP [203], should be considered. The key benefits of such refactored TCP stacks are:

    - No kernel modifications needed;
    - Eliminated expensive system call overhead by translating system calls into Inter-Process Communication (IPC);
    - Batched event handling (function calls) and batched packet Input/Output (I/O).

    The TCP connection engine is just one part of efficiency optimization problem. Another, more complex issue is the support of CC functionality, and in this context, the number of concurrent TCP connections with a particular CC algorithm that can be generated from a test platform would greatly depend on the CC algorithm we want to support, since each CC has a set of algorithm-specific associated parameters/variables (as it was detailed in Section 2.3), which would require storage in each per-connection TCB. Hence, there is a performance-scalability trade-off that needs to be made.

3. CUBIC's sensitivity to different packet loss patterns is closely associated with another critical aspect: the accuracy of latency (RTT) measurements under heavy packet loss, which directly affects the RTO adjustment (due to Karn's algorithm) and goodput/throughput estimation. One of the solutions to this problem could be to use a recommended TCP ***Timestamps*** option [90]. Even though the specification suggests that, according to recent studies, using the Timestamp option to collect more RTT samples does not lead to a better RTT estimator for modest CWND sizes, we argue that for high-speed connections with large CWND (e.g., CWND > 15000 segments) this is a necessary mechanism to account for retransmission of bursts of lost packets, during which there, otherwise, would be prolonged idle periods without any RTT/RTO updates during the recovery time. In addition, in a large CWND regime on high BDP paths, this option is useful in protecting against the wrapped-around sequence numbers problem (e.g., 2^{32} sequence number space can wrap around multiple times within a few minutes of a long-lived connection), by using it with the ***Protection Against Wrapped Sequences (PAWS)*** mechanism [90]. Recommendations to improve the RTO adjustment procedure, provided in the latest specification update [204] only suggest reducing the initial RTO value to 1 s (from 3 s by default) for low-RTT networks, and the use of finer system clock granularity (e.g., $<$ 100 ms, instead of default 500 ms) is reported to provide "somewhat" more accurate estimation results. However, optimization of ***alpha*** (RTT gain) and ***beta*** (deviation gain) parameters is an open research issue.

Packet Loss Recovery Efficiency of CUBIC flows

This part of our research activities targeted ***Packet Loss Recovery Efficiency (PLRE)*** as a collective performance indicator (a set of metrics), comprised of several components, such as: Loss Recovery Duration, time-average Goodput and Throughput, Goodput/Throughput ratio as well as the number of retransmissions (due to $FRTX/FREC$ and RTO) and recovery events (per connection), in a network environment with a dominant synchronous packet loss pattern in a Drop-Tail (FIFO) network buffer scenario. We have considered the impact of different combinations of loss recovery and detection algorithms/heuristics on the aforementioned performance characteristics of high speed long-lived (not application-limited) TCP CUBIC connections, sharing the bottleneck resources with standard TCP flows (Reno and NewReno). The following scenarios were tested:

1. Standard ***FRTX/FREC***(from NewReno) only;
2. ***FRTX*** + ***Selective ACKnowledgement (SACK)*** (RFC3517 SACK-based loss recovery);
3. ***FRTX*** + ***SACK*** (RFC3517) + ***Limited Transmit (LT)***;
4. ***FRTX*** + ***SACK*** (RFC3517) + ***Duplicate-Selective Acknowledgment (DSACK)*** + ***LT***;

5. ***SACK* + *Proportional Rate Reduction (PRR)*** (2 versions) + ***LT***;
6. ***SACK* + *RFC6675***-based enhancement + ***LT***.

The purpose of this study was to evaluate how specific algorithms interact with each other within the loss recovery context, and whether any specific combination of these algorithms can actually improve or worsen the loss recovery characteristics of high speed CUBIC flows. The operational network conditions were varied as follows: 1) RTT range: 40 ms - 140 ms; 2) Bottleneck buffer size: 25% - 100% of a BDP; 3) Bottleneck bandwidth: 50 Mbit/s - 1000 Mbit/s; 4) random loss: fixed $BER = 10^{-12}$. The following observations were made:

1. ***Bottleneck router state.*** With a FIFO queueing discipline in the router, the bottleneck queue size is increasing with the increase of the output buffer size (25% - 100%) as expected; however, considering the loss recovery algorithms tested, the largest impact on the queue size is observed from the ***FRTX* + *SACK*** (RFC3517) combination, while other extensions have no significant influence. This pair of algorithms produces the largest queue increase factor under moderate BDP (40 ms, 50 Mbit/s), ranging from +44% to +20% in the buffer size range of 25% - 100% of BDP, respectively, as compared to using only standard ***FRTX/FREC*** mechanism. Queueing delay follows the same dependency pattern, with the increase factor ranging between +37% - +19%. There are two reasons for that:
   - The number of buffer overflows (i.e., dropped packets) follows an inverse pattern, decreasing with the increase of the buffer size, since more packets are accommodated at the cost of increased queueing delays;
   - This combination of algorithms produces higher intrinsic loss rate, since SACK-based algorithm allows recovering targeted blocks of lost data faster before a chain of RTO events, which would lead to waves of queue drain effects, reducing the buffer pressure, but at the expense of goodput drop.
2. ***Fast Retransmit/Recovery duration.*** Loss recovery times are largely affected by the queueing delay inflation, and may range from $2 \times RTT$ to $4 \times RTT$ (per recovery session), under 25% - 100% buffer size scheme, respectively, and a moderate BDP (40 ms, 50 Mbit/s). The major impact on the evolution of this metric, again comes from the ***FRTX* + *SACK*** (RFC3517) combination, where RFC3517-based recovery introduces 40% - 80% (under 25% - 100% buffer sizing scheme) improvement in loss recovery time for CUBIC flows, and around 80% for NewReno flows disregard the buffer size setting.
3. ***Goodput/Throughput ratio, time-average Goodput.*** This metric is used to measure the proportion of "useful" data bytes (not including any retransmissions) as compared to the overall volume of transmitted bytes (both original and retransmitted). This ratio linearly increases with the increase of the buffer size (leading

to a lower packet loss rate). It has been observed that when scenario (4) with the ***DSACK*** extension is activated, the effective goodput of NewReno flows drops by 0.15% in small buffer regime (25% of the BDP), but leads to increase of goodput of TCP CUBIC connections by 0.15%, on average. In addition, enabling the ***LT*** algorithm results in a goodput drop for NewReno flows by 0.5% - 0.8%, while increasing the goodput of CUBIC flows by 0.1 %, on average. As regards to the average goodput, the ***SACK*** (RFC3517) extension, when enabled, provides the highest goodput gain (39% - 12% under 25% - 100% buffer sizes, respectively) for CUBIC flows, compared to any other combinations of extensions without SACK-/RFC3517. However, the flows, which are joining the shared network later (e.g., 20 s and 100 s after the simulation start) cannot fully converge to a fair share with the initial CUBIC flows (Sender1 - Receiver1). Another observation is that in the moderate BDP regime (40 ms, 50 Mbit/s), CUBIC flows may obtain up to 50% lower goodput compared to NewReno flows and up to 19% of Reno flows; the detected root cause of that behavior is the TCP-friendly mode/algorithm of CUBIC, which, based on the measurements, may produce lower estimated CWND size value for a standard flow modeled (tracked) within a TCP CUBIC algorithm itself, which results in lower average CWND sizes of CUBIC than that of competing real NewReno flows. Such behavior disappears in the larger BDP setting (e.g., BW=500 Mbit/s and RTT=90 ms).

4. ***Number of initiated FRTX/FREC events; retransmissions due to FRTX/FREC and RTO events.*** In general, the number of retransmission (all types) events is decreasing with the increase of the bottleneck buffer, as we discussed earlier. When the TCP SACK-based recovery mechanism (RFC3517) is enabled, the number of packets retransmitted via the ***FRTX*** mechanism for CUBIC and Reno flows is 250% - 180% (under 25% - 100% buffer sizing, respectively) higher compared to the case when RFC3517-based recovery is not used. This allows reducing the number of expensive RTO events 30-120 times, but does not affect significantly the number of RTO-based retransmission sessions for CUBIC flows. However, enabling the ***LT*** algorithm together with SACK/RFC3517 allows reducing the number of RTO events by 6% - 12% for CUBIC and NewReno flows.

5. ***Proportional Rate Reduction; SACK-based RFC6675 recovery.*** Recent research studies proposed new mechanisms to enhance packet loss recovery, namely Google's ***PRR*** [119][126] extension and RFC6675-based recovery [115](an improved RFC3517 SACK-based mechanism).

The described performance matters, including the PRR and SACK-based RFC6675 extensions, are being currently further investigated and the results are targeting a journal publication.

The presented analysis suggests that the available algorithms and heuristics for packet loss detection and recovery may not be as accurate and fast in detecting and reacting to the

packet loss, with the biggest challenge being to accurately distinguish a real packet loss from reordering, as well as unnecessary retransmission of packets, which were erroneously marked as lost when an associated RTO event triggered their retransmission. In these cases the sender would receive 3 consecutive DACKs, but they are not necessarily an indication of a new congestion instance. In addition, there are reported communication corner cases, when some of the tested algorithms/heuristics may fail [91][120]. As a result, alternative mechanisms, such as ***Recent Acknowledgement (RACK)*** [205][206], have been recently proposed, which rely on packet transmission timing information rather than the packet counts or order to infer a packet loss (for both original and retransmitted packets) quicker and more accurately (according to the initial reported testing results) than the currently deployed solutions. What is more, a new TCP functional architecture, called TCP ***Laminar*** [207][208], has been introduced, which separates the CC (the amount to send) functionality from the transmission scheduling (when to send) actions into two independent subsystems and introduce a set of new state variables to provide a convenient mapping between the existing and new CC algorithms. In fact, this is an attempt to perform a code "refactoring" of the existing CC standards to allow for easier development and testing of new algorithms by eliminating a heavy dependency of multiple deployed algorithms on the same set of CC variables (CWND, SSTHOLD, etc.), which makes it very difficult to make any algorithm-specific changes [207]. This note concludes the discussion of the TCP CC mechanisms, investigated in this part of the research project.

2.5 Chapter Summary

In this chapter, the importance and evolution of TCP CC (Congestion Control) mechanisms for high speed Internet communication has been discussed, introducing the key building blocks and individual algorithms, which together form a framework to enhance the stability and reduce the possibility of major congestion collapses in the global distributed Internet. The main operational states of a TCP connection, including the connection establishment, data transmission phase as well as connection termination step are briefly introduced in order to emphasize the architectural complexity of the TCP protocol and associated supporting mechanisms with a large number of various intermediate steps and interactions between its functional components. The chapter further presents a detailed and comprehensive overview of the evolution of the CC mechanisms from the early days of the Internet until now. This discussion is further extended with an analysis of different high speed CC approaches, including a set of algorithms, which were deployed on an Internet-wide scale and used as the default CC control extensions, such as Compound TCP, TCP CUBIC and other experimental extensions. This part of the chapter allows forming a better insight on the complexity of the Transport layer (4) CC in the context of high speed communication networks, such as the Internet. The reason is that the modern high speed adaptive CC control algorithms can be categorized into 4 main groups, such as Loss-based, Delay-based, Hybrid Loss-Delay-based and Network model-/congestion-based algorithms. The first two categories exhibit very different operational behaviour in terms of reaction to the changes in the network state, namely Loss-based algorithms treat

packet loss as an indication of congestion and back-off the transmission rate when the congestion has presumably occurred, while Delay-based solutions use delay as a proactive way to predict a possible congestion build-up. On the other hand, hybrid solutions use both approaches to control the transmission rate. The problem arises when different types of CC algorithms are sharing a communication path, where Loss-based solutions, such as TCP CUBIC, which is a current default CC mechanism in Linux and Windows 10 OS, outperform other variants by using more aggressive and scalable mechanism, namely a CUBIC function profile and a set of scaling parameters, to regulate its Congestion Window (CWND) growth. This part of the discussion is finalized by introducing an alternative new CC approach, developed by and deployed by Google, namely the TCP BBR, which belongs to a category of network-model-based algorithms, and is shown to be significantly more efficient in terms or lower induced packet queueing latency. Thus, the analysis in this chapter shows that the network CC complexity arises as a result of combination of multiple critically important factors, such as the heterogeneity of modern high-speed networks and developed TCP protocol extensions. This statement leads to the main research objectives of this work, which are comprised of performance evaluation of high speed TCP CUBIC connections in networks with large Bandwidth-Delay Product (BDP) of the paths. Two studies consider such communication aspects as: algorithmic robustness properties of high speed TCP CUBIC connections with large CWND size in networks with a random packet loss profile, and PLRE of high speed CUBIC connections considering the impact of various combinations of loss detection and recovery algorithms, coupled with CC functionality. The conducted studies are important due to the following reasons: 1) they further assess the operational stability of currently massively deployed CC algorithm under difficult operational network conditions, which may be encountered, e.g., in mixed network environments, such as fixed-wireless environments; 2) the PLRE study allows assessing the impact of synchronized packet loss on flows with large CWND sizes that will define such operational metric as Flow Completion Time (FCT), which estimates the duration of data transfers (e.g., distribution of large volumes of experimental data, i.e. Big Data flows), etc.

CHAPTER 3

Testing and Performance Characterization of Data Center Networks

In the age of global digitalization, ***Data Centers (DCs)*** have become mission critical infrastructures, with a goal of providing the means to process, store and distribute large volumes of digital data, produced by a multitude of different applications and services with diverse performance and scale requirements. As as a result, a large number of new DC facilities are being built all over the world to accommodate constantly growing service deployment (both end-user and business-centric) and operational demands, as well as to bring the provided web-scale services closer to users, residing in geographically distributed areas. According to the latest Cisco Global Cloud Index (GCI) report [17], global IP traffic, crossing the Internet and IP ***Wide Area Network (WAN)*** networks is estimated to reach 3.3 Zetta Bytes (ZB) per year by 2021; however, global DC traffic was already estimated to be 6.8 ZB/year in 2016, and this volume is projected to ***triple*** by 2021, reaching tremendous ***20.6 ZB/year*** (nearly 95% of which will be cloud-based traffic). This estimate includes DC-to-user (14.9%), DC-to-DC (13.6%) and intra-DC (71.5%) or "East-West" traffic; however, the total "East-West" traffic may reach around 85%, if we add rack-local (stays within a given server rack) traffic [17], and due to this, the distribution of traffic volumes will change, accounting for more than 90% of intra-DC traffic. These numbers clearly indicate that, over the years, the DC traffic patterns have changed significantly, shifting from a classical client-server based communication model with monolithic application/service components to more complex, modular and multi-tier service architectures and application programming models, such as the MapReduce framework [20], requiring interaction of multiple distributed components for application service delivery. However, this is just a tip of the iceberg among a wide range of factors, which have facilitated the emergence of new DCN performance requirements, creating a need for fundamental architectural and operational transformation of the DCs and their supporting network interconnects. The most notable factors, which are challenging the DC design and operation dogmas, and facilitate adoption of new technologies and paradigms, are summarized as follows [17][209][210][211][212][213]:

1. ***Cloud service models***. Fast adoption of the Cloud Computing paradigm [21] and Cloud service models [21][209], such as Software-as-a-Service (SaaS), Platform-

as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS) or generalized Anything-as-a-Service (XaaS), resulted in the emergence of new data-intensive, mixed traffic profiles, which traditional DC architectures were not designed for. On the other hand, this created a range of new opportunities for the enterprises to scale and transform their business activities and to deploy new types of business applications and services in a more flexible and resource-efficient way [210]. Three main categories of cloud infrastructure models [211][17] are currently in use, namely public, private (on-prem or collocation) and hybrid cloud offerings, where public cloud is gaining momentum and is expected to become the dominant choice (near 73% of all installed workloads) for the enterprise segment by 2021 [17].

2. ***Hyperscale DCs***. Web-scale service providers, such as Amazon (AWS), Google, Facebook, Microsoft, Apple and alike have been the main drivers of the DC architectural transformation and are usually the first adopters of the latest software and hardware technologies, aiming to continuously optimize their disaggregated DCNs, focusing on the best operational practices in the automation, availability, reliability, energy efficiency and scalability (economy of scale principle) aspects. The latest projections show that the Hyperscale DCs will be hosting around 53% of all the DC servers and process up to 55% of the global DC traffic by 2021 [17].

3. ***Virtualization technologies***. Fast proliferation of virtualization and "modularization" technologies [17][212], ranging from the host-level solutions (Virtual Machine (VM), hypervisors, containers) to ***Network Virtualization (NV)*** (e.g., overlays, network slicing) and advanced resource orchestration and management approaches (e.g., micro-segmentation). A new concept of ***micro-services*** is being actively adopted and promises future-proof approaches for building highly modular service and application architectures with independently extensible components, greatly reducing the development and deployment time.

4. ***SDN/NFV, spine-leaf architectures***. Operational experience from the design, deployment and maintenance of Hyperscale DCs [38][39][214][215][216][217][218] showed that the combination of a flattened 2-3 tier ***Spine-Leaf*** DC architecture with ***Software Defined Networking (SDN)*** based control and orchestration framework and additional automation tools, can provide the required flexibility and operational efficiency, and enables a "scale-out" approach for DC expansion to support increasing bandwidth, storage and processing demands. Furthermore, the SDN paradigm is actively being coupled with the ***Network Function Virtualization (NFV)*** concept, allowing to decouple any network function (e.g., switching, routing, firewall, load balancing, etc.) from a dedicated physical device, and run such network functions (in a form of a ***Virtual Network Function (VNF)*** in software) on commodity hardware (servers). According to estimates, the SDN/NFV functional elements will be transporting up to 44% of intra-DC traffic by 2021, and the number of deployed DCs with full or partial SDN adoption may reach 67% on a global scale [17].

5. ***IoT, Big Data***. Massive volumes of data, generated within and outside of DCs, have facilitated the development of new data analytics approaches, commonly referred to as ***Big Data Analytics*** [219] allowing to efficiently process large volumes of data and extract useful information, which can be used to improve the provided services and business processes, diagnose operational anomalies and security threats, develop new scientific models and much more. Apart from that, a new wave of global digitalization and automation, collectively described as fourth industrial revolution (Industry 4.0 standard) [220] as well as efforts to create a global distributed network of massive number of connected devices, conceptualized as the ***Internet of Things (IoT)*** or ***Internet of Everything (IoE)***, creates fundamentally new challenges and requirements for DCN performance and architectural scalability. These factors have facilitated active research of more efficient ways of data aggregation, distribution and processing, and the current focus of the research and industrial community is on ***micro DCs*** and ***Edge DCs*** [221][222], aiming to bring the compute and storage resources closer to the data sources to offload the centralized cloud DCs.

The cumulative effect of the outlined factors and trends on the evolution of DCs and their performance assessment is the following:

1. Complex interactions between different associated components of multi-tier cloud applications and services, including internal Big Data Analytics and other parallel processing systems, SDN/NFV technologies, multi-level virtualization techniques, hybrid cloud deployment models (e.g., critical business apps in the enterprise cloud, and customer front-end components in the public cloud) all lead to highly variable, complex traffic profiles, which make the DC design and planning, as well as traffic growth prediction, a very challenging task.

2. Consequently, new technologies and approaches, which are continuously being developed for DCNs, must be thoroughly tested before a production deployment. Moreover, considering the scale and complexity of modern DCs (even on an enterprise level), building and operating a DC may incur tremendous financial expenditures (CApital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX)), and any wrong decision in the design, planning, implementation/deployment steps may have severe operational consequences and business impact.

3. In addition to that, in many cases, new ideas and academic research results may never be transferred to real-life production DC deployments, because the performance evaluation results and conclusions may not be applicable at scale due to limited research resources (test infrastructure), too many critical assumptions and simplifications introduced in the applied research methodology to compensate lack of large-scale experimental facilities or complexity of replicating real services.

4. As a result, there is a strong need to develop a comprehensive DCN testing and performance evaluation framework, which would allow conducting a diverse set

of realistic performance characterization experiments, incorporating the aforementioned technologies, realistic test applications/services and networking effects (variable operational conditions, real- or near-real-time processing, etc.). In addition, one of the most critical requirements of such testing is that it should (must) be applicable at larger scale in order to be able to project/apply the obtained results to real DCN deployments, but without incurring large associated financial costs and not compromising the accuracy of the obtained test/measurement results.

Last but not least, massive growth of the number and scale of new DC facilities, which is projected to continue [17], has raised serious concerns about the amount of energy consumed by these facilities [223][224][225], and that created a need to consolidate the DC industry efforts to develop more efficient DC design, deployment and operational practices (with active input from the web-scale companies, like Facebook or Google) and to define energy efficiency estimation approaches. One of such industrial collaboration efforts is known as The Green Grid association [226], which makes notable contributions in the form of new energy-usage-related performance metrics, best practice recommendations, e.g., a widely used ***Power Usage Effectiveness (PUE)*** metric, a new ***Performance Indicator (PI)*** metric [227], free cooling maps and other increasingly important open standards. Therefore, DC ***Energy Efficiency (EE)*** must be considered as a DC performance evaluation component, which is as important as traditional performance indicators, such as throughput, latency or availability of the DC IT infrastructure, and should be incorporated into the overall DCN performance evaluation and testing strategy in order to better reflect the operational efficiency of a DCN.

This introduction and problem statement forms the agenda for the subsequent discussion. The main focus of this chapter is on the following three DCN-oriented research problems investigated, such as: 1) scalable DCN testing approaches; 2) the impact of SDN on the performance and testing of DCNs; 3) Energy Efficiency in DCNs with optical switching (a complementary study). A more detailed analysis and results can be found in the related papers, included as part of this thesis.

The outline of this chapter is as follows: in Section 3.1, a summary of a methodology for scalable DCN testing using a hybrid physical-simulated testbed and up-to-date DC, SDN/NFV standardization and best practices as well as testing guidelines are discussed. A study of Energy Efficiency benefits of exploiting optical switching in DCNs is briefly discussed in Section 3.2. Section 3.3 presents the analysis of the advantages and challenges of using SDN in DCNs, including the testing and performance evaluation perspective. Finally, the chapter is summarized in Section 3.5.

3.1 Scalable testing methodology for DCNs

3.1.1 Current state in DCN testing: new initiatives and challenges

In order to better understand the context and challenges, associated with large scale (and high performance) testing of DCNs, this Section highlights the key up to date standardization activities, covering the DC design aspects, as well as technologies and

frameworks, which are being actively integrated in DCN environments. A summary of the most notable standardization and current best practice developments is provided in Table 3.1.

Please note that the provided list is not exhaustive (more information about other SDN-related standardization efforts can be found in [228]) and the purpose of the following overview is to highlight the strategic importance of all the diverse efforts to identify and formalize the key requirements and solutions for DC/DCN evolution, and, ultimately, to stress the need for a comprehensive testing and performance benchmarking framework.

As it can be seen in Table 3.1, three main categories of some form of standardization activities around DC facilities, SDN/NFV, DC-related open source projects as

Table 3.1: Data Center, SDN and NFV standardization activities, open source initiatives and testing/performance benchmarking guidelines

Institutional standardization	
Standard, specification, recommendation or Working Group	**Institution**
Data Center Benchmarking Methodology, RFC 8239	**IETF**
Data Center Benchmarking Terminology, RFC 8238	IETF
Benchmarking Methodology for Network Interconnect Devices, RFC 2544	IETF
Benchmarking Methodology for LAN Switching Devices, RFC 2889	IETF
Methodology for IP Multicast Benchmarking, RFC 3918	IETF
Software Defined Networking Research Group (SDNRG)	IRTF (IETF)
Network Function Virtualization Research Group (NFVRG)	IRTF (IETF)
ANSI/TIA-942-A Infrastructure Standard for Data Centers	ANSI/TIA
ANSI/BICSI 002-2014 Data Center Design and Implementation Best Practices	ANSI/BICSI
CENELEC EN 50173-5 Information Technology - Generic Cabling Systems Part 5: Data Centres	CELENEC
ISO/IEC 24764 Information technology - Generic Cabling Systems for Data Centres	ISO/IEC
Joint Coordination Activity on Software Defined Networks, JCA-SDN-D-001 Rev.6	ITU-T
P1916.1 - Standard for Software Defined Networking and Network Function Virtualization Performance	IEEE
Software Defined Virtual Networks (SDVN) project, Internet and Scalable Systems Metrology Group	**NIST**
Industry Specification Group for NFV	ETSI
Tier Standard: Topology, Operational Sustainability	Uptime Institute
OpenFlow Specifications, OpenFlow Testing specifications	ONF
Open source projects, industrial collaboration	
Project, framework	**Foundation, association**
Open Compute Project (OCP)	OCP Foundation
Linux Foundation Networking Fund: FD.io, OpenDaylight, ONAP, OPNFV projects	The Linux Foundation
ONOS, CORD, XOS, Mininet, Stratum, Trellis, etc.	ONF
The Green Grid	The Green Grid Association
Industry-driven, web-scale best-practice (custom) deployments	
Facebook Fabric, FBOSS, Wedge, Back-Pack, 6-pack, etc.	Facebook
Saturn, Jupiter DCN architectures	Google
Brain-Slug BGP SDN for large-scale DCNs	Microsoft
Acronyms	
IETF (Internet Engineering Task Force), IRTF (Internet Research Task Force), ANSI (American National Standards Institute), TIA (Telecommunication Industry Association), BISCI (Building Industry Consulting Services International), CELENEC (European Committee for Electrotechnical Standardization), ISO (International Organization for Standardization), IEC (International Electrotechnical Commission), ITU-T (Telecommunication Standardization Sector of the International Telecommunications Union), NIST (National Institute of Standards and Technology), ETSI (European Telecommunications Standards Institute), ONF (Open Networking Foundation), FD (Fast Data), ONAP (Open Network Automation Platform), OPNFV (Open Platform for NFV), ONOS (Open Network Operating System), CORD (Central Office Redesigned as Datacenter), XOS (Extended Operating System), FBOSS (Facebook Open Switching System), BGP (Border Gateway Protocol).	

well as custom industrial solutions, can be distinguished, namely the ***organizational standardization*** (performed by major regional/international standards bodies, such as IEEE, IETF, ETSI, ANSI and alike), ***industrial collaboration*** (consortiums, associations) in a form of ***open source projects*** and ***custom DC-oriented architectural solutions***, shared (partly or fully open-sourced) by the web-scale giants, which have greatly influenced the development of common best practices, especially in the cloud-DC deployment aspects. The first category is comprised of ***Standards Development Organizations (SDOs)***, whose key focus is on DC infrastructure design and implementation [229][230], including DC-tier-defined infrastructure design [231], physical DC design aspects (cabling, racks, power distribution, etc.) [230][231][232][233], network device-level benchmarking recommendations [47][48][234][235][236], and SDN/NFV related standardization [228][237][238][239][240][241], including the OpenFlow specifications [242]. Note that in this category either specific identified standards, which focus on the DC-related evolution, or formed Working Groups within specified organizations are presented, which started actively formulating SDN/NFV (including DCN use-cases) standardization roadmaps in the recent years. The second category is very important, since industry-driven collaborations are positioned as actual "de facto" drivers of faster adoption and practical deployment of the latest technologies, as compared to long standardization processes that increase the time-to-market and hinder fast innovation. Leading industrial collaboration examples include Open Compute Project [243], focusing on the promotion and development of open hardware architectures for DCNs, The Linux Foundation [244], hosting a variety of open source networking software projects (OSs, platforms, etc.), Open Networking Foundation (ONF) [245], which, in addition to the OpenFlow standardization, is a non-profit network-operator-led consortium working on defining new transformation strategies for carrier-grade networks; further, The Green Grid association [226], working on the best practice solutions for energy efficiency in DCs. Finally, the third category represents the web-scale service providers, which have developed valuable expertise in building, optimizing and operating cloud-scale global intra- and inter-DC infrastructures, and started sharing some of their designs and methodologies. Some of the most well-known solutions include Facebook Fabric [246] for DC interconnect and Dynamo DC power management system [218], modular DC switch designs (Wedge, BackPack, 6-Pack) and a switching platform (FBOSS) [247]; Google's large-scale DC architecture designs [39] or Microsoft's Brain-Slug BGP-based SDN DC architecture [248] provide a valuable technological insight as well.

The resulting conclusion from this discussion of the standardization and best-practice frameworks is that, while SDN, NFV and cloudification of the DC infrastructures was aimed at reducing the operational complexity and associated costs, increasing the agility and flexibility of the IT business models, as well as addressing the automation, performance, and scalability needs, there is still lack of practical understanding of how all these innovative, but scattered in different projects and platforms, technologies can benefit one or the other enterprise or a businees process. Some technical analysts indicate [249] that the current-state SDN/NFV standards do not define new operating practices to match

these new technologies yet; hence, the true benefits of change are not addressed properly. One of the main reasons of that is that there is lack of comprehensive technical testing and performance benchmarking practice to properly assess the advantages of one new solution over another. Analysis of the discussed standardization and best practice efforts (presented in Table 3.1) shows that, at the time of writing, there are only a few identified DCN testing and performance benchmarking specifications with practical recommendations of how to conduct different types of tests, namely defined by IETF in [48] and NIST in [240]. However, in [48] specification (informational), the guidelines are targeting benchmarking of isolated network devices (Device Under Test (DUT)), rather than specific network configurations, technologies or topologies. In fact, [48] document mainly details the tests, described in earlier documents (RFC 2544 [47] and RFC 2889 [236]), and there are some examples of "chained" device benchmarking, for specific technology tests (IP multicast) though, defined in [235], but these "standards" are suitable only for obtaining theoretical performance indications and small lab tests, as argued in [48] and [250]. On the other hand, initiatives outlined by the NIST's Internet and Scalable Systems Metrology Group in [240], are promising an answer to some of these testing questions, since there are concrete defined activities, targeting the development of research and measurement science (how-to) for SDN/NFV, building a distributed physical-emulated testbed for SDN/NFV studies, also in the DCN context, as well as new test tools and deployment guidance. However, this is still a work in progress, and there are not many details available yet.

Finally, the most important DCN testing issue is that available test frameworks and recommendations are limiting the scope to testing/benchmarking of individual network devices or functional elements (e.g., a SDN controller, virtual switches, a specific protocol or a service, etc.) separately, while there is an obvious need for a comprehensive ***Integrated System Test (IST)***, where the cumulative impact of different components (devices, virtualization technologies, service models, protocols, etc.), operating at different layers, can be accurately assessed at scale, larger than a typical lab setup with a few available physical devices to test. This is illustrated in Figure 3.1. A conceptually similar ***IST*** approach is well-defined [230] for the stress-testing of a physical-electrical-mechanical and ***Heating, Ventilation and Air Conditioting (HVAC)*** infrastructure of a DC, and commonly referred to as the ***level five DC commissioning process (IST)*** or "integration phase" [251], which allows to stress-test an operational DC facility at its maximum capacity/load, when all the related systems are connected. This test allows ensuring that the DC can operate in the busiest conditions, and make timely adjustments after identifying the problems at the very beginning, rather than disrupting the production DC with running services.

The discussion, presented in this Sub-section, highlights the existing gap between the theoretical frameworks and yet ongoing standardization efforts, and the practical understanding of how diverse technologies, multi-layer architectures and related functional components of DCNs can be tested in a unified/integrated way and realistic communication context. An attempt to tackle this challenge is presented in the next Sub-section.

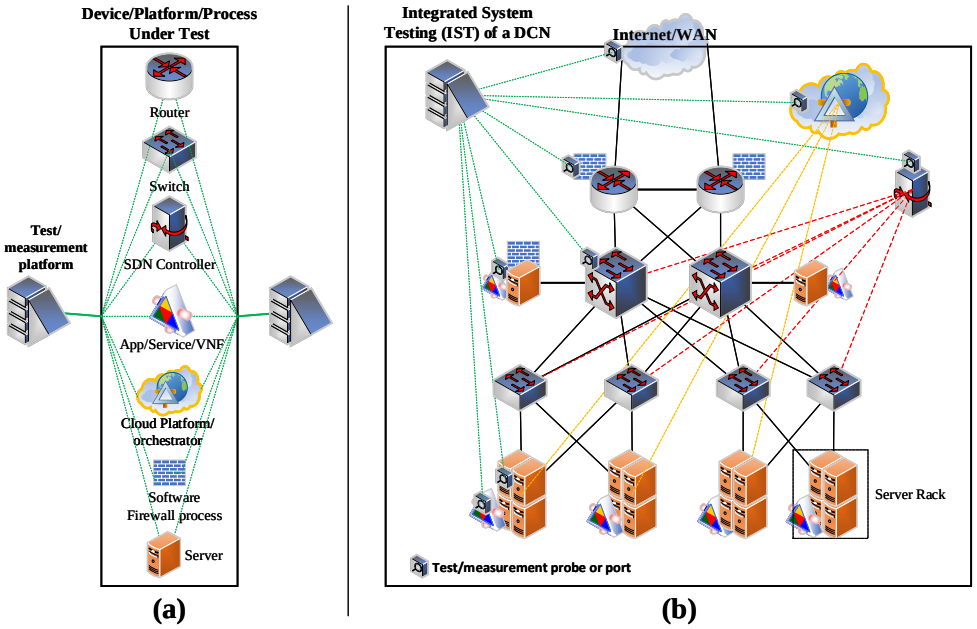


Figure 3.1: (a) Isolated testing versus (b) Integrated System Testing approach

3.1.2 Large-scale Data Center Network testing approach

This Sub-section briefly presents the key research findings and observations, following from the conducted research activities, presented in a set of papers, included in the second part of this thesis, namely ***Paper D*** - ***Paper F***, ***Paper H*** and ***Paper I***, as well as places the work into perspective compared to other testing methodologies.

3.1.2.1 Hybrid physical-simulated optical-electrical testbed

The work, presented in ***Paper D***, is a Proof-of-Concept (PoC) study, which has laid the foundation for subsequent studies and enhancements of a hybrid physical-simulated DCN testbed. The main pursued goal was to investigate a flexible methodology for Data Center Network performance and scalability studies, which would allow achieving the following DCN-oriented testing objectives:

1. Be able to carry out comprehensive research activities at scale, while not necessarily having access to a real production scale DCN and the application/service workloads and not incurring tremendous CApital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX).

2. Possibility to evaluate different DCN topologies. The defined methodology should not be restricted by the availability of physical DCN hardware, so that a wide range

of different architectural solutions can be tested both separately and simultaneously.

3. Retain some realistic network processing effects, e.g., packet queueing/switching delays, transient network performance variation due to physical network/transmission properties, firmware/software implementation, etc.

4. Support realistic application/service models or deployment of real applications/services. As it was emphasized at the beginning of this Chapter 3, real Data Center applications and services, both legacy and Cloud-based, are built using complex architectural relations between different functional components, which produce not uniform, but rather bursty, dynamic and changing over time traffic profiles, so that these interactions cannot be realistically modelled or represented by basic statistical distributions, such as Poisson, uniform or Pareto.

5. Stable environment, possibility to conduct repeatable experiments with reproducible results. This is a critical requirement for the verification and validation process of the obtained results.

6. Support of virtualization, software-defined networking and/or cloud-related functionality. The "softwarization" trend in DCNs dictates a strong need to incorporate Software Defined Networking (SDN)/Network Function Virtualization (NFV) and Cloud-related aspects into the overall DCN testing framework.

7. Experiment/task automation capabilities. This is a critical factor allowing to focus on the research objectives rather than performing time-consuming element-by-element re-configuration of the test network and experimental parameters upon the need.

These objectives were approached by analyzing the state-of-the-art work in the area of scalable and high performance DCN testing and evaluation. The conducted analysis shows that there have been various strategies trying to address this aspect, such as:

- Extending/building simulation tools for large-scale DCN modelling and performance studies, with the most notable ones being the CloudSim toolkit [53][252] and a broad range of its extensions [253][254][255][256][257], and other similar tools [258][259].
- Hybrid hardware-software based testbeds. In this case, researchers attempt to build high performance, programmable DCN testbeds in two ways:
  - ***Field-Programmable Gate Array (FPGA)*** based solutions. For example, interconnecting multiple FGPA boards with Network Interface Card (NIC) cards into arrays of programmable high performance SDN-controlled switches, directly interconnected with multiple ARM processors [260]; such switches can be used for both computing and networking functions, which can be distributed to both ARM processors or FGPAs. Another similar example

is presented in [261], but in this case FGPAs are combined to form a high performance processing grid structure, where DC components are simulated as abstracted execution-driven performance models on top of Linux kernel, rather than real network protocol stacks and functions. However, in a large-scale study in [261] authors conclude that:

  * Such combined FPGA-based solutions need massive simulation power even when evaluating the DC processing at the rack level.
  * FPGAs are quite slow and provided tools (FGPA Velirog/systemverilog) are not productive. Hence better tools are needed.
  * Massive scale simulations are error-prone with transient software errors and crashes being the most significant reasons of discrepancies in the research results obtained.
  * As regards to scale factor, performance evaluation results obtained for small-medium scale setups cannot be generalized to large-scale.

– ***Commodity hardware combined with emulation tools, SDN and orchestration platforms***. This is one of the first attempts to combine the OpenStack [262] infrastructure, a SDN controller and a scalable network emulator, called Common Open Research Emulator (CORE). The network devices are emulated in CORE using virtual switches and OS-level virtualization (based on Linux containers).

As regards to the Cloud-scale testing activities conducted by the industry giants, such as Microsoft or Google, they follow two mixed approaches. For example, Microsoft recently disclosed that they run a virtual copy of the entire Azure Network infrastructure in a simulation [263] in Amazon Web Services (AWS) Cloud on Elastic Compute Cloud (EC2) instances on 500 Virtual Machines. Google, on the other hand, actively uses production scale test DC facilities [264]. There are also strategic government-supported initiatives, such as the Swedish RISE SICS [265] large-scale DC facility dedicated for research, education and testing purposes. However, this is not a feasible option to replicate a DC facility just for testing purposes for most of the researchers, even from Microsoft's point of view [263].

As regards to simulation-based approach, these tools mostly support either detailed modelling of the Application layer functionality, but very limited or no network layer consideration (such as mentioned CloudSim), while other frameworks provide advanced capabilities for modelling of network interconnect [54], but with no functional application/service layer support, limited to the traffic generation with a few statistical distributions (Poisson/exponential, uniform, Pareto, Weibull, log-normal, etc.) or simple application traffic profiles, mimicking video, audio or data traffic.

Based on the defined research objectives and discussed limitations of other solutions, in this work, a hybrid physical-simulated testbed was created and, following the PoC study, introduced earlier, instead of modelling the entire network interconnect in a sim-

ulation/emulation environment, real physical DCN switches, both with electrical and all-optical switching capabilities were combined with modelled DCN infrastructure, as described in more detail in the associated ***Paper E***. As stated earlier, the need for greater degree of network programmability and an efficient control framework, facilitated the investigation of possibilities to add SDN-based control to the testbed setup in order to enable a unified control framework for diverse DCN components (physical electrical and optical switches, simulated switches), integrated together, as presented in ***Paper F***.

A subset of the latest testbed's performance evaluation results are briefly described next in order to highlight the current state of the hybrid system, and potential applicability for scalable DCN research and experimentation. Specifically, the potential performance gains of using parallel kernel simulations on a multi-core system as compared to the standard sequential kernel simulations, where the entire simulation model and packet processing is carried out using a single thread and only one Central Processing Unit (CPU) core, are discussed.

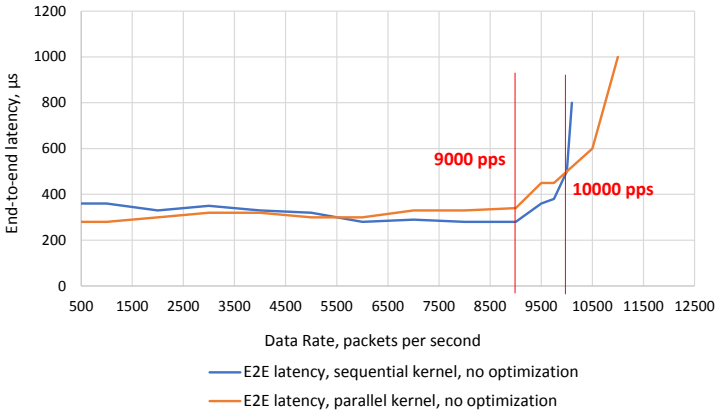


Figure 3.2: End-to-end communication delay with sequential and parallel simulation kernels in a single-switch test scenario

The results, presented in Fig. 3.2, show the evolution of the end-to-end communication latency as a function of traffic load in packets per second (a Maximum Transmission Unit (MTU) of 1500 bytes is used) for two simulation kernel modes tested, namely the sequential and the parallel, but without the OS kernel tuning and optimization in this scenario. It can be seen that with the parallel kernel there is a relative latency reduction at the low packet rates, while sequential kernel exhibits better stability throughout a range of packet rates. However, parallel kernel mode allows to handle higher system load (in the range of 10000 to 10500 packets per second (pps)) than in the sequential kernel. In general, the conclusion that can be made based on the observations from the measurements

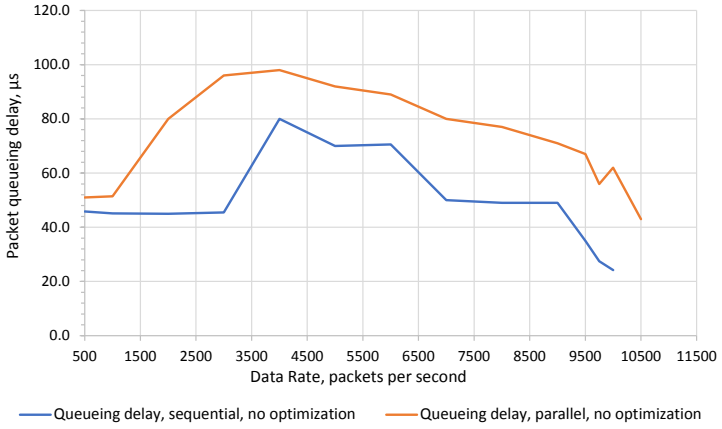


Figure 3.3: Queueing delay at the SITL gateway with sequential and parallel simulation kernels in a single-switch test scenario

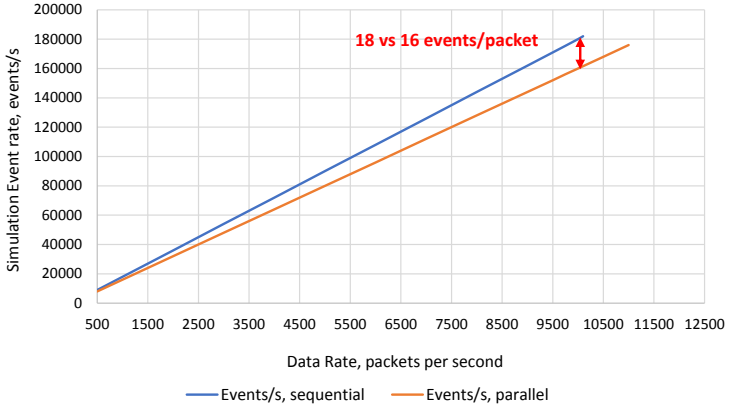


Figure 3.4: Simulation event rate (events/s) with sequential and parallel simulation kernels in a single-switch test scenario

is that the simulation system with real-time packet processing is able to handle traffic loads as high as 9000 pps (109.29 Mbit/s) in sequential kernel and around 10000 pps (121.44 Mbit/s) in parallel kernel, while exhibiting reasonable performance in terms of delay variation/jitter, packet queueing delays and the overall stability. Fig. 3.3 shows the evolution of the queueing delays at the virtual System-in-the-Loop (SITL) interface, and the incurred packet queueing due to buffering is higher when using parallel kernel. One of the reasons of such difference between the two modes is that in the parallel kernel mode,

the core of the Modeller software tries to distribute the packet processing load evenly via available reserved N+1 cores, where N is the number of SITL gateway nodes used in the modelled network, and the scheduling process is being regularly interrupted by the kernel of the underlying OS (Linux CentOS kernel was used) to serve other system tasks. From the results provided in Fig. 3.4 it can be seen that the parallel simulation mode results in lower simulation event rate for the same processed traffic load, with the average number of events per processed packet being 16 events/packet against 18 in the sequential case. This suggests that parallel kernel can handle higher system loads, and this behavior is observed when a set of OS kernel, NIC buffer and scheduling parameters are tuned.

There has been an extensive effort to optimize the performance of the simulation framework for more efficient and stable real-time operation, primarily targeting different OS-level performance optimization profiles, when a UNIX-like system (Linux CentOS 7.3 and Ubuntu 18.04 servers were used in our setup) is used as a host for the simulation tool. Each considered performance profile (please check [266][267] for a reference) constitutes a set of interrelated tweaks of the operational parameters (e.g., modifying kernel packet buffer size (backlog queue), process scheduling priorities, size of the NIC's ring buffer, managing the Interrupt Coalescence or disabling the CPU power saving states (C-states), etc.), producing relatively optimal performance, which must be chosen carefully, since the outcome may be very use-case-specific. Fig. 3.5 shows that, among all the tested performance profiles, the ***Latency Performance (LP)*** profile allows achieving the most stable and the lowest end-to-end intra-simulation latency when a Real-Time kernel (RTk) of an OS is applied (below 400 µs for 95% of collected latency samples, and below 520 µs for 98%-tile of the samples), however, at the cost of relatively higher packet queueing delays at the SITL interface, as it can be seen in Fig. 3.6 (around 90 µs for 98% of collected latency samples, against < 80 µs with just LP profile used without the RT kernel).

This matter, namely the system optimization/tuning aspects, is currently being further investigated and results thereof are targeting a journal publication.

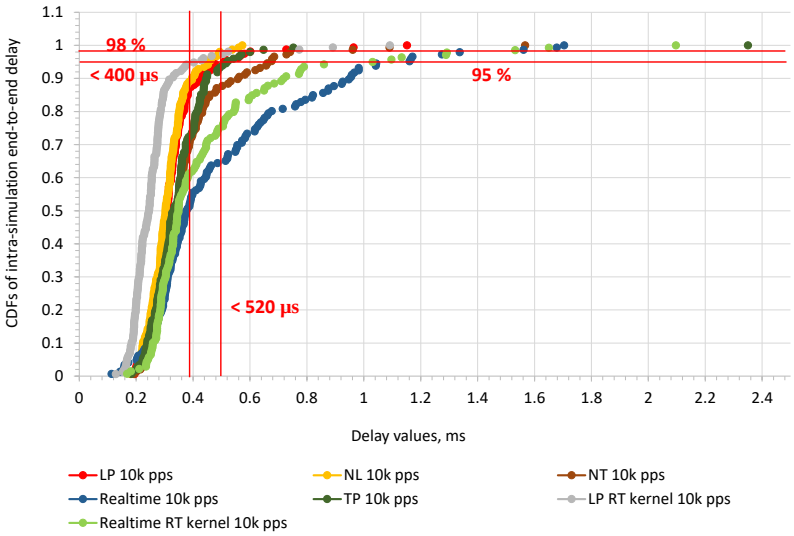


Figure 3.5: Performance tuning results with parallel kernel (N+1 cores, 10k pps load): end-to-end intra-simulation latency. Performance profiles: LP - Latency Performance, NL - Network Latency, NT - Network Throughput, TP - Throughput Performance, RT- Real Time OS kernel (Linux CentOS), CDF - Cumulative Distribution Function. Note: 10k = 10000

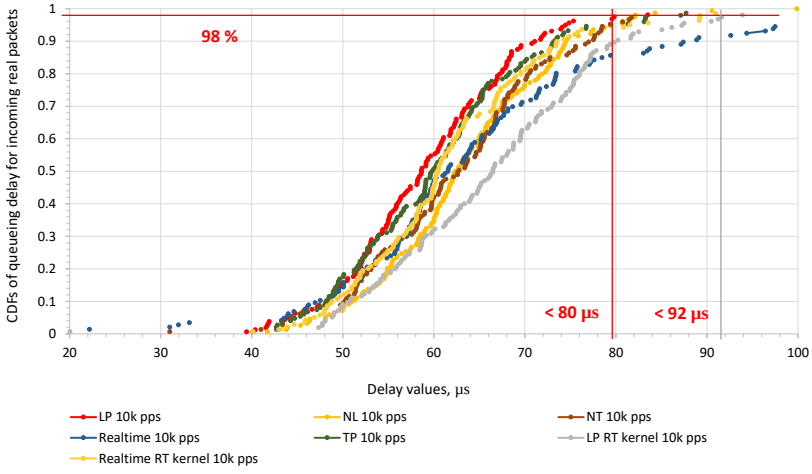


Figure 3.6: Performance tuning results with parallel kernel (N+1 cores, 10k pps load): queueing latency at the SITL virtual interface

3.2 Data Center Energy Efficiency as a performance criteria

A complementary study of Energy Efficiency in DCNs with optical switching, selectively applied on different network layers of three considered DCN topologies, such as a traditional Tree, a Fat-Tree and a Ring-based structure, is presented in a related ***Paper E***. It is generally known that DC facilities and the HVAC systems, as well as servers in particular, are consuming significant electrical energy resources [225][268]. In this work the main goal was to assess whether such optical technologies as Wavelength Division Multiplexing (WDM) can be effectively applied to DCNs from the power consumption point of view. Effectiveness of such traffic aggregation approaches as Traffic Grooming is evaluated, which provides benefits of efficient data packing into potentially fewer optical wavelength resources, thus, requiring fewer active ports on the associated network devices. For more details and better insight of the conducted research study, please refer to the corresponding paper (i.e., ***Paper E***).

3.3 Software Defined Networking paradigm: benefits and testing challenges

Throughout this Ph.D. thesis it has been emphasized that the ***SDN*** paradigm, since its initial introduction in a set of research projects and further evolution into highly flexible, feature-rich and functional implementations and deployments, was projected as a future-proof solution for any imaginable networking problem. The present-day functional capabilities, provided by different SDN-related software platforms and protocols, including fast proliferation of programmable test solutions into the optical domain, is unarguably driving the transformation and evolution of the operational aspects of DCNs. Nevertheless, such rapid "softwarization" of network infrastructures, while offloading the control/signalling tasks from the data plane, at the same time introduces new challenges, since all the complexity is now being shifted to the software realm. That dictates the need for more efficient software testing approaches, as well as functional testing and performance evaluation of such SDN-scalability-related aspects as flow-rule installation efficiency, full exploitation of the Traffic Engineering (TE) capabilities and other areas, requiring additional investigation. Thus, in this part of the work, SDN performance aspect is considered in two contexts: 1) Optimization-analysis of the Flow rule distribution efficiency in SDN-enabled switches in ***Paper G***, as well as 2) exploration of SDN-based TE prospects, detailed in ***Paper H***. Additionally, a case-study exploring the traffic generation and performance measurement capabilities of hardware-based network testers in SDN-enabled DCNs, is provided in ***Paper I***.

3.4 Large-scale Integrated Network Testing: a comprehensive view

The potential and applicability of a hybrid physical-simulated electrical-optical and SDN-enabled DCN testbed has been investigated in this work, as it is detailed in Section 3.1, in addition to other research probes. The discussion of this Section focuses on the potential

and prospects of high-performance, scalable and comprehensive DCN evaluation and testing from a broader perspective, and how this can be achieved in practice by combining different available open source software tools.

The open research questions in scalable DCN testing, covered so far in this work, constitute an integral and key important component in a DCN research area. Nevertheless, in addition to the methodology on how to scale DCN infrastructure for research purposes, it is also important to consider the following aspects, which become critically important when dealing with performance evaluation and testing at scale:

1. ***Dynamic test network parameter tuning, configuration management and automation***. Changing the configuration of just a few network devices manually may not seem to be a time-consuming and error-prone task, but considering the scale factor, e.g., if a large network topology needs to be configured and tested, some form of network configuration automation and management must be considered to reduce the re-configuration time and to have a unified view of the network configuration state. For this purpose, if we consider the hybrid SDN-enabled test network infrastructure, introduced in Section 3.1.2, network configuration can be managed via any up-to-date SDN control framework, such as ONOS [269], OpenDaylight [270], Ryu [271] or other, using OpenFlow Management and Configuration Protocol (OF-Config) [272] and/or NETwork CONFiguration (NETCONF) [273] protocols. Otherwise, if a testbed setup includes legacy/non-SDN network devices, rConfig [274] or other configuration management framework may be used. As regards to scalable server infrastructure management, e.g., physical/VM-based servers, such tools as Chef [275], Puppet [276], Ansible [277] or other can be exploited for this task.

2. ***Scalable network performance monitoring and visualization***. This functionality is particularly useful for real-time performance observations (e.g., live dashboards) as well as performance studies over a period of time, based on the collected historical performance data (logs). This strategy allows offloading time-consuming and inefficient manual statistics processing workload to a set of tools, highly optimized for such tasks. While this can be achieved by the means of SDN functionality (via statistics polling), in large-scale test networks with, e.g., hundreds of network nodes and a large set of monitored metrics, this may not be very efficient, since the SDN control plane, in addition to the key network control tasks, will need to store and process potentially large-volumes of statistical data in real-time, and that may negatively affect the SDN controller's performance. Hence, alternative powerful solutions can be used, such as: Nagios Core [278], OpenNMS [279], Zabbix [280] or other monitoring tools, as well as Cacti [281], grafana [282] or statEngine [283] visualization platforms. There are also promising possibilities to gather real-time power consumption data from network devices (per-module) via Simple Network Management Protocol (SNMP) [284][285].

3. ***Test application/service design***. Generic traffic generation tests with uniform or Poisson traffic profiles to stress-test a network are not sufficient for any realistic performance characterization, since it is widely known that the traffic patterns commonly observed in DC environments are bursty, with high peak-to-average load ratio due to the architectural nature of the applications and services deployed (multi-tier, parallelized processing, complex inter-component interactions). Therefore, it is critically important to be able to produce as realistic test application/service behaviour as possible, in order to obtain a more accurate indication of the impact of, e.g., application/service scaling (to serve more client service requests) on the stability and performance of a Network Under Test (NUT), or other factors. This task can be accomplished by using a powerful Juju [286] application design and deployment platform, which allows composing advanced application models, assembled from, e.g., Docker [287] or Kubernetes [288] containers (software containerization platforms).

4. ***Statistics collection, data storage and processing***. High-performance and large-scale DCN testing, depending on the research objectives, may produce a large volume of statistical data, which needs to be collected and stored for subsequent processing (unless a stream-based real-time processing strategy is used) and analysis of the obtained results in a fast and resource-efficient manner. There is a number of powerful open source tools, such as High-Performance Computing Cluster (HPCC) [289], Hadoop [290], Apache Spark [291], Storm [292] and others, which employ massive parallel/batch, usually a cluster-based, data processing principle by distributing the processing tasks to multiple compute nodes, working in parallel, or use other fast Analytics methods as in MapD [293]. This approach allows one to significantly increase the data (e.g. gathered statistics) processing efficiency and reduce the required time. For data storage, such database tools as InfluxDB (time-series database) [294], HDF5 [295], MongoDB [296], HBase [297] and other high performance scalable solutions can be used. Efficient network statistics collection can be achieved with graphite [298], Telegraf [299] or sFlow [300][301], or host-level statistics with Host sFlow [302].

5. ***Machine Learning/Deep Learning, Analytics***. Machine Learning (ML) and Deep Learning (DL) frameworks are becoming important due to advanced data analysis capabilities, allowing to model and predict the evolution of network or service performance over time and obtained results can be used for timely network parameter tuning for testing purposes or in real deployments. Therefore, such frameworks will become a vital part of any comprehensive DCN performance evaluation methodology and should be considered as a useful mechanism in high performance network testing. A broad range of fast open source frameworks are already available, such as TensorFlow [303], SciKit-Learn [304], MCT (CNTK) [305], MXNet [306], or statEngine [283] for real-time data Analytics being the most notable ones.

Fig. 3.7 presents a high-level view on how a highly functional and customizable Integrated Network Performance evaluation and testing architectural framework may be structured, considering the functional capabilities of the available technologies and new Open Source software tools and platforms, which provide great opportunities for creating an agile and multi-functional DCN performance characterization and testing framework for conducting comprehensive and high quality research, while not incurring tremendous financial costs of deploying such a system. In Fig. 3.7, purple frame boxes indicate the areas investigated in this work and emphasize how they link to other approaches in an integrated context. This schematic drawing emphasizes two important aspects, such as:

- Integrated DC Network Testing approach in a high-performance and large-scale context is a complex research challenge, covering a large number of interrelated aspects, such as traffic generation and performance measurement accuracy, network scaling and network configuration, efficiency of data collection/processing and the presentation of results, etc.
- The operational DCN aspects have become greatly dependent on the functional capabilities of the software, including the automation and virtualization. Despite all the associated scalable DCN testing challenges, new open source software tools and platforms greatly expand the functional testing capabilities, allowing to build a comprehensive performance characterization framework, and this can accomplished incrementally, in a modular, independent fashion.

This discussion finalizes the chapter, and a summary of the key research findings and observations is provided in the next Section.

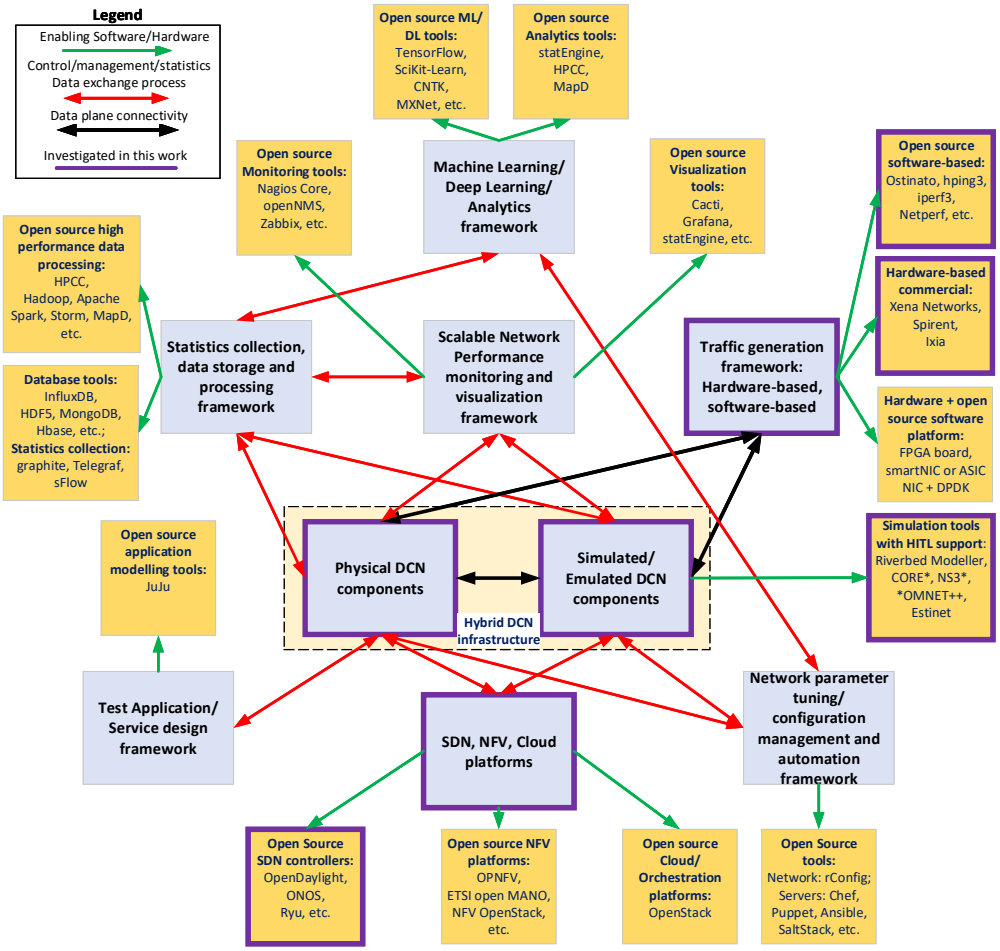


Figure 3.7: A vision of Integrated Network Testing: new opportunities. Note: * - limited multi-core parallel processing support

3.5 Chapter Summary

This chapter begins the discussion by providing a comprehensive overview of the main factors affecting and seriously challenging the DC operational and design principles, the most notable ones being fast proliferation of Cloud service models, network virtualization technologies, architectural DCN transformation influenced by the SDN/NFV paradigms, as well as the impact of tremendous traffic growth in a form of Big Data. This introduction sets the direction for further discussion by emphasizing that all these new technological paradigms and complex multi-tier applications and service deployment models lead to

bursty, dynamic traffic profiles, which significantly complicate any DC planning or traffic growth prediction. The main problem highlighted in this context is that building and operating a DCN may incur high financial costs, and due to this reason it is generally unfeasible to build such a facility just for testing and research purposes. Therefore, a controversial situation arises, when there is a strong need to test new innovative technological solutions for DCNs aimed at reducing the operational complexity and optimizing the performance, while not being generally able to test these approaches in a realistic DCN production environment at scale. In order to provide a better insight of the DCN testing challenges, up-to-date standardization efforts, covering SDN, NFV, industrial best-practice and testing guidelines are discussed, with the main conclusion being that most of the standardization activities with practically applicable solutions are still at the very early stages of development, and an Integrated Network Testing approach being a missing component. A promising and innovative approach to tackle this complex matter of scalable DCN testing and performance characterization, namely a testing methodology, focusing on creating a functional DCN testbed, is further introduced in this chapter. The DCN-oriented testing objectives are defined first, followed by the analysis of the state-of-the-art attempts of the research community to tackle this complex challenge, so that the introduced testbed architecture is set into a perspective against these available solutions. The DCN testbed, presented in this work, constitutes a hybrid physical-simulated, electrical-optical and SDN-enabled solution for active experimentation. A subset of obtained testbed's performance evaluation results, detailed in the associated publications, suggest that this system can be used for further exploration of SDN-enabled testing and experimentation, while requiring additional performance tuning and optimization, reducing the dependency on the OS kernel space limitations in particular. This chapter further highlights the work conducted in the context of SDN-oriented testing and performance characterization, namely a flow-rule placement algorithm for SDN switches and analysis of Traffic Engineering possibilities provided by SDN in DCNs. A complementary DCN energy efficiency study, focusing on the impact on WDM-enhanced optical switching on power consumption in a set of DCN topoligies, is highlighted as well. The last part of this chapter presents a high level view and discussion on how a comprehensive Integrated Network Testing approach can be realized in practice by exploiting the functional capabilities of open source software. It also puts the overall work, conducted during this part of the Ph.D. project, into perspective and forms a broader view on additional large-scale testing aspects that require research attention.

CHAPTER 4

Conclusions

The topic of network performance evaluation and testing, especially in the context of high speed, is becoming critically important due to the rapid expansion of the global communication network infrastructure, including the traffic-intensive Internet and the Data Center Networks (DCNs). On one hand side, the deployed and new communication networks need to be scaled in order to accommodate constantly increasing traffic volumes as a result of two main factors, such as the increase of the global Internet user-base, as well as fast proliferation of new, feature-rich applications and services, with Internet-based video streaming/downloading and Big Data Analytics being the most notable ones in terms of the volume. On the other hand, the underlying network infrastructure provides the resources, such as network bandwidth, for the upper layer protocols of the communication stack, necessary to deliver the requested service/application data from one communication end-point to the other. However, the algorithmic properties of these protocols or their specific components define how efficiently these network resources will be used. Therefore, such properties must be thoroughly studied, documented and, if necessary, even modified in a timely manner in order to eliminate or alleviate any potential communication instability problems in a global context. Furthermore, scalable and high performance network testing is necessary to better understand and assess the impact of new developed communication principles, technologies, architectures and services on the evolution of the operational state of a realistic network environment, but large-scale testing resources are not always available. As a result, the following identified open research questions within the discussed context have been investigated in this Ph.D. thesis: analysis of the algorithmic properties of high speed persistent Transmission Control Protocol (TCP) CUBIC connections focusing on Congestion Control (CC) in terms of algorithmic stability and Packet Loss Recovery Efficiency (PLRE) in a network environment with random and synchronized packet loss patterns, respectively. Furthermore, the scalability and performance aspects of DCN testing are analyzed in the context of a proposed hybrid testbed; additionally, SDN-related testing aspects as well as DCN energy efficiency questions in the context of optical switching are investigated.

Statistically, it is known that TCP provides reliable end-to-end data transmission service to over 90 % of the global Internet Protocol (IP) traffic on the Internet. The complexity in the context of high speed and especially long-distance communication arises as a result of the fact that the generic TCP/IP protocol stack, which has been deployed for several decades, wasn't initially designed to handle such loads and diversity of traffic profiles. Therefore, as regards to the Transport layer (4) of the communication

protocol stack, a large number of different algorithmic fixes and extensions, including the CC algorithms, have been coupled with the TCP connection engine over the years to provide more stable and scalable end-to-end connection experience. However, the truth is that there is still no common agreement and lack of practical insight with regard to which algorithmic extensions, when combined, provide the most optimal and smooth network communication experience. Therefore, in the first part of this research work, presented in Chapter 2, a protocol-algorithmic level performance evaluation study of multiple high-speed TCP CUBIC connections is conducted in two different communication scenarios. A comprehensive theoretical analysis, preceding this performance study, provides a detailed overview of the most relevant properties of TCP protocol and its main associated extensions, high speed CC algorithms in particular, which allows a deeper insight of the underlying high speed CC complexity. The first considered scenario evaluates the impact of high random packet loss and increasing Round Trip Time (RTT) of the flows in a shared network environment on the throughput stability of CUBIC flows. The outcome of this study suggests that high speed CUBIC flows with large Congestion Window (CWND) sizes are affected more in terms of the relative throughput (as well as congestion window) reduction than flows with smaller CWND sizes in a shared environment, with the main trend being that the average throughput of high speed flows gradually decreases with the increase of the experienced end-to-end delay in the moderate packet loss region ($BER = 10^{-9}$) even with scalable cubic profile, while another effect is observed under higher packet loss ($BER = 10^{7} - 10^{5}$), where cubic flows attempt to recover the throughput in larger leaps with the increase of the RTT of these flows. PLRE study in the second scenario suggests that high speed CUBIC connections with activated Fast Retransmit, SACK-based (RFC3517) recovery and Limited Transmit (LT) algorithms exhibit the highest degree of communication stability and shorter induced loss recovery duration times, as compared to other investigated combinations. This study allows assessing the impact of synchronized packet loss (due to the network buffer overflow states) on the flows with large CWND sizes, as this aspect defines such operational metric as Flow Completion Time (FCT), which estimates the duration of large data transfers (e.g., distribution of large experimental data, i.e. Big Data flows).

DCN performance characterization and testing at scale has been discussed in Chapter 3 of this part of the thesis. The identified scalable DCN testing objectives, based on the analysis of the state-of-the-art solutions with their benefits and limitations, suggest a set of criteria, which was used to develop a methodology of scalable DCN testing. A related preceding analysis of the up-to-date standardization efforts in the DC design and operation, SDN, NFV and available testing recommendations indicates that this is still an open research area, and most of the DCN-related performance benchmarking strategies propose some guidelines of testing of isolated network devices or systems (Device/System Under Test), whereas some first attempts to define a comprehensive Integrated System/Network Testing methodology, such as initiated by ETSI and NIST organizations, are work in progress with no practical solutions available yet. Furthermore, the discussed state-of-the-art research initiatives, presented in a form of DCN testbeds,

which can be grouped into simulation-based and hybrid physical-simulated/emulated, both have pros and cons, which must be carefully considered when choosing a large-scale DCN testing methodology. For instance, some of the available simulation tools, designed for large-scale performance studies, achieve this scalability in DCN performance characterization at the cost of oversimplified network and application layer modelling using performance models and software abstractions. Therefore, such relative simulation scalability is undermined by the missing network layer functionality (interaction of real network protocols) and processing effects (buffering, forwarding, transmission specifics). As regards to the second category of physical-simulated/emulated approaches, most of the earlier studies rely on FPGA based testbed setups, but some of these solutions, which showed high performance gains in FPGA-based network grids, are modelling the network interconnects as abstracted performance models with limited protocol stack support (if any). This factor may hinder the applicability of such setups to realistic network and protocol scalability studies. The testing approach, presented in this work, is also classified as hybrid physical-simulated, but the main difference is that several different connectivity options have been tested, such as: modelling the entire DCN in the simulation and attaching high performance hardware-based traffic generators to stress-test the system; interconnecting real physical DCN switches with the simulated ones to form a hybrid DCN topology, while retaining some realistic network processing effects in the physical part of the setup; the last stage of testbed's enhancement attempted the integration of the previous setup of physical-simulated network switches with an external SDN control plane in order to form a unified SDN-based control framework. The performance evaluation results suggest that, in its current state, this type of hybrid setup can be used for active experimentation and testing of SDN-related communication aspects, such as stress-testing of the control plane request processing capacity and other scalability aspects. The maximum achievable real packet processing rate of around 9000 pps in the sequential simulation kernel mode and 10000 pps in the parallel mode, which is equivalent to approximately 109.29 Mbit/s and 121.44 Mbit/s with an MTU of 1500 bytes, suggests that at the moment this setup may not be suitable for large-scale data plane transmission tests, and additional investigation of such system-level performance aspects as kernel space interrupt scheduling, interrupt coalescence, and isolation binding of logical CPU cores to the modeller software process and virtual SITL gateway nodes, could potentially improve the situation. In this case, since the OS kernel space (or kernel-libpcap driver interface in Linux) is identified as the limiting processing factor of the testbed to allow for higher packet processing rates, alternative ways of porting the simulation environment to, e.g., a Data Plane Development Kit (DPDK) based environment to eliminate the OS kernel-space processing, can be further considered.

Since the SDN paradigm has become the cornerstone of the overall network "softwarization" strategy, driving faster adoption of new software-based solutions and redefining the DCN operational principles and dogmas, testing and performance characterization aspects in relation to SDN were investigated in this work as well, and are covered in Chapter 3. However, it has been noted in Chapter 1 that, despite all the hot discussions

and activities around DC architectural evolution and the need for a technological shift through the SDN realm, the current DC and SDN/NFV standardization efforts are yet failing to properly address the true benefits of adopting SDN and transforming the existing non-SDN infrastructures to enable a smooth integration of SDN-based platforms and solutions into the existing ecosystem. The main factors of concern delaying the decision of migration to a fully SDN-based architecture for the enterprises are security (immature implementations, lack of successful use-cases with addressed security concerns, etc.), cost of deployment and lack of technical expertise to navigate through this new technological environment. Therefore, this discussion suggests that there are various open research questions, which require more thorough consideration in the SDN context. The focus of two conducted SDN-related performance studies was on the network-device-level testing, encompassing performance characterization of a Flow rule placement algorithm with the overall goal to increase the Flow rule capacity in the SDN switch by performing intelligent migration of flows between hardware and software implementations of the Flow rule storage table in the switch. This study emphasizes the benefits of such an approach (increased number of accommodated Flow rules) and exposes the transient software processing effects in the form of latency spikes for the packets of flows, migrated from hardware to software. The second study is focusing on the analysis of Traffic Engineering capabilities, enabled by the use of SDN in DCNs. The main idea of this work is to provide a concise yet comprehensive overview of the key observations with regard to SDN TE applied in DCNs, exposed in multiple extensive literature surveys. The experimental part of this work is intended to show-case the experimental activities, conducted as part of this Ph.D. project and highlight the role of the scalable hybrid testbed in large-scale SDN TE testing activities.

Last but not least, the energy efficiency study, conducted by the means of network-level simulations and dimensioning of three different investigated DCN topologies, namely a three-tier Tree, a Fat-Tree and a ring structure with optical circuit switching, selectively applied at different network layers and most importantly, enhanced by the WDM capabilities, presents interesting results in the light of extensive efforts of the DCN research community to introduce optical switching in the DCNs. The main observations are that the benefits of optical switching, when the power consumption is of concern, may very much depend on which particular layers of specific topologies the optical WDM-based switching is applied. Therefore, the power consumption level varies from architecture to architecture on different layers. Overall, optical switching applied in the aggregation or core and aggregation yields relatively the best results. However, it's very important to take into consideration other critically important factors, such as the traffic profiles, scale of the topology modelled, as well as traffic grooming strategy applied (end-to-end or hop-by-hop). To conclude, the presented power consumption results are of indicative nature, showing a relative power consumption level as compared to the general case, when a traditional Tree topology is used without optical switching.

Part II

Included Papers

Paper A/Poster: High Speed Network Evaluation and Testing

A. Pilimon, "High Speed Network Evaluation and Testing", in *5th PhD School on Traffic Monitoring and Analysis (TMA)*, Poster Presentation, Barcelona, Spain, 21-22 April, 2015.

DOI:

High Performance Network Evaluation and Testing

A. Pilimon, PhD student, E-mail: artpil@fotonik.dtu.dk
Department of Photonics Engineering, Technical University of Denmark

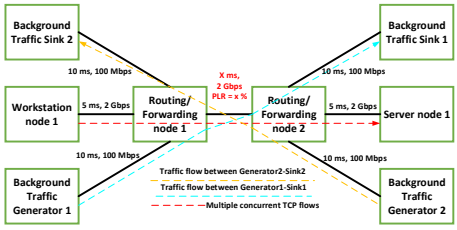
5th Traffic Monitoring and Analysis (TMA)
PhD School 21-22.04.2015
Barcelona, Spain

High Speed TCP connections for performance evaluation

- A major fraction of today's globally generated Internet traffic relies on TCP as a transport protocol, which plays a critical role in ensuring a reliable and connection-oriented transport service (Layer 4).
- Feature-rich and bandwidth-intensive applications (financial/high-frequency trading data, large volumes of experimental research data, multimedia content) require optimized, scalable and efficient algorithms and communication protocols. Efficient TCP Congestion control solutions are of utmost importance.
- As a result, proactive performance evaluation tests must be leveraged in order to characterize the state of the network under different operating conditions (varying workloads, application-specific requirements). Therefore, it is important to ensure that the defined testing scenarios are as close to reality as possible.
- Considering the global Internet as such, a large fraction of servers are Linux-based (successful integration of OpenStack impacts the choice as well) and a currently default system-wide TCP Congestion control algorithm in Linux is a high-speed TCP CUBIC.
- Extensive experimental and simulation-based analysis of this algorithm showed that, despite all the benefits (high scalability, stability), there are several serious drawbacks (controversial RTT- and TCP-fairness, poor performance in wireless environments, loss synchronization). In addition to that, available results are often contradicting .

Complementary simulation-based study of TCP CUBIC

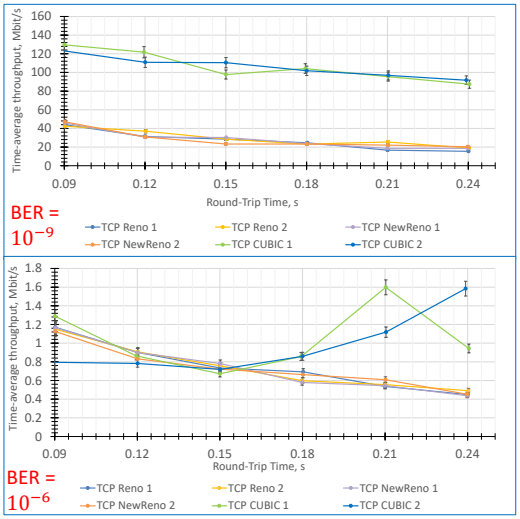
- A simulation-based analysis of TCP CUBIC was conducted in our work in order to gather additional results and to get an insight of the adaptation capabilities and operating robustness of multiple TCP CUBIC connections under severe operating conditions.
- The simulation network setup is a popular dumbbell topology, widely used for analysis of the congestion control algorithms. We considered the following scenarios:
  1. Multiple different long-lived TCP connections (TCP CUBIC, Reno, NewReno) in a network environment with a large BDP (Bandwidth-Delay Product), increasing random packet loss rate and increasing RTT (Round-Trip-Time) of the flows.
  2. Multiple different long-lived TCP connections in a network with a bottleneck transit link (variable bandwidth), different buffer sizes at the bottleneck and increasing network load (increasing number of TCP CUBIC flows, joining the network). Drop-Tail buffer is used.
- Future work: Evaluation of the impact of SACK, Limited Transmit and ACK Heuristics algorithms on the packet loss recovery efficiency of high-speed TCP CUBIC flows in the network environment with a synchronized packet loss pattern (due to a bottleneck link).



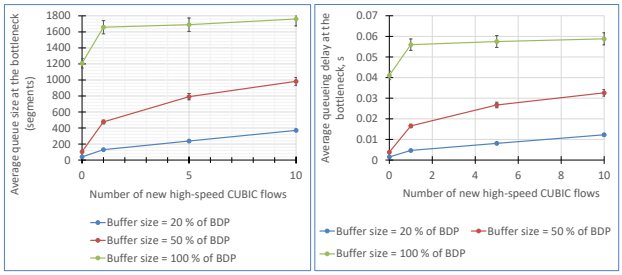
A logical scheme of a Simulation setup

Results

Scenario 1



Scenario 2



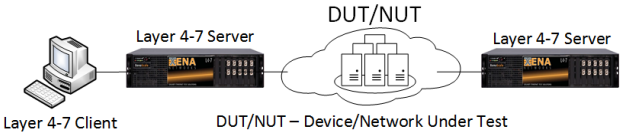
Discussion

- A non-linear and highly scalable window increase algorithm of CUBIC leads to a transmission of large packet bursts in the network environment with random loss, but a constant (deterministic) loss probability *p*, with *1/p* packets between two successive loss events. Fast increase of the transmission rate leads to a larger number of randomly dropped packets within a period of time.
- Loss-based approach of congestion control in CUBIC and a loss-free-time-dependent congestion window growth rate, maintain the utilization of the buffer at the maximum level. New high-speed CUBIC flows contribute to the build-up of large packet queues and increased queueing delays.
- Different buffer size settings at the bottleneck node (Router) may alleviate the performance degradation, but at the cost of decreased throughput. Hence, more advanced queue management algorithms will improve the situation to a certain extent.

Large scale network testing: challenges and opportunities

When it comes to real-life network testing as such, in order to be able to perform a large-scale network performance evaluation, there are a few available strategies, but the most realistic one is a emulation of a large number of concurrent TCP connections through a scalable testing platform. Since each established TCP connection, especially when the high-speed TCP flavours are used, maintains relatively large amount of connection state information (TCP Connection Block), significant amount of memory and computational resources might be consumed. If we consider emulation of millions of simultaneous TCP connections (a realistic scenario for a medium-sized backbone or a Data Center Network), multiple scalability problems of a network tester may arise. Thus, several alternative implementations of a TCP protocol stack have been considered, which support only the generic functionality of a TCP connection engine (requiring much less connection state information). An example of such a simplified TCP connection engine is a light-weight-IP (lwIP) stack. While it can be deployed for testing purposes, there are still several important questions to answer: will it be sufficient enough for a realistic network evaluation, and what is a scalability penalty for a high-speed TCP connection testing approach in a resource-constrained tester? These are the questions we would like to answer as a future step in our research. We have considered a L4-7 network testing platform, provided by Xena Networks ApS, as a potential environment for experimental prototyping. It offers the following testing capabilities:

- Scalable Gigabit TCP testing (1 G, 10G and 40G interfaces)
- Stateful Traffic generation (load) with 24 M TCP Clients and 24 M Servers on one platform
- Connection ramp up rate: 12 M connections/s.



Research topics of current interest

Data Center Networks

SDN Energy Efficiency

The following is a listing of **Paper A/Poster: High Speed Network Evaluation and Testing**, formatted for presentation clarity.

5th Traffic Monitoring and Analysis (TMA) PhD School 21-22.04.2015, Barcelona, Spain

High Performance Network Evaluation and Testing

A. Pilimon, PhD student, E-mail: artpil@fotonik.dtu.dk

Department of Photonics Engineering, Technical University of Denmark

High Speed TCP connections for performance evaluation

- A major fraction of today's globally generated Internet traffic relies on TCP as a transport protocol, which plays a critical role in ensuring a reliable and connection-oriented transport service (Layer 4).
- Feature-rich and bandwidth-intensive applications (financial/high-frequency trading data, large volumes of experimental research data, multimedia content) require optimized, scalable and efficient algorithms and communication protocols. Efficient TCP Congestion control solutions are of utmost importance.
- As a result, proactive performance evaluation tests must be leveraged in order to characterize the state of the network under different operating conditions (varying workloads, application-specific requirements). Therefore, it is important to ensure that the defined testing scenarios are as close to reality as possible.
- Considering the global Internet as such, a large fraction of servers are Linux-based (successful integration of OpenStack impacts the choice as well) and a currently default system-wide TCP Congestion control algorithm in Linux is a high-speed TCP CUBIC.
- Extensive experimental and simulation-based analysis of this algorithm showed that, despite all the benefits (high scalability, stability), there are several serious drawbacks (controversial RTT- and TCP-fairness, poor performance in wireless environments, loss synchronization). In addition to that, available results are often contradicting.

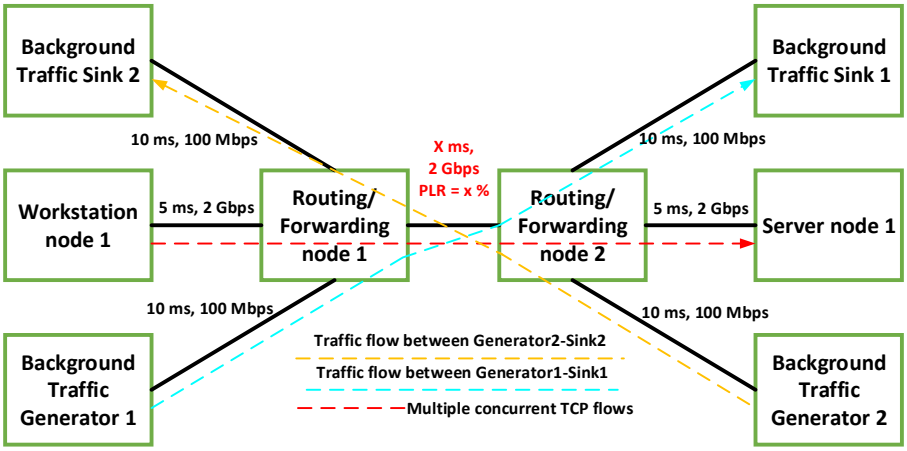
Complementary simulation-based study of TCP CUBIC

- A simulation-based analysis of TCP CUBIC was conducted in our work in order to gather additional results and to get an insight of the adaptation capabilities and operating robustness of multiple TCP CUBIC connections under severe operating conditions.
- The simulation network setup is a popular dumbbell topology, widely used for analysis of the congestion control algorithms. We considered the following scenarios:

    1. Multiple different long-lived TCP connections (TCP CUBIC, Reno, NewReno) in a network environment with a large BDP (Bandwidth-Delay Product), increasing random packet loss rate and increasing RTT (Round-Trip-Time) of the flows.

    2. Multiple different long-lived TCP connections in a network with a bottleneck transit link (variable bandwidth), different buffer sizes at the bottleneck and increasing network load (increasing number of TCP CUBIC flows, joining the network). Drop-Tail buffer is used.

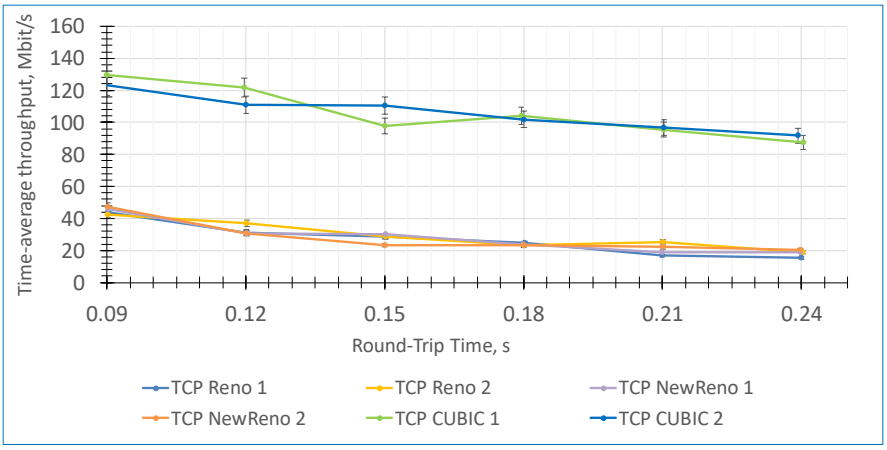
- Future work: Evaluation of the impact of SACK, Limited Transmit and ACK Heuristics algorithms on the packet loss recovery efficiency of high-speed TCP CUBIC flows in the network environment with a synchronized packet loss pattern (due to a bottleneck link).

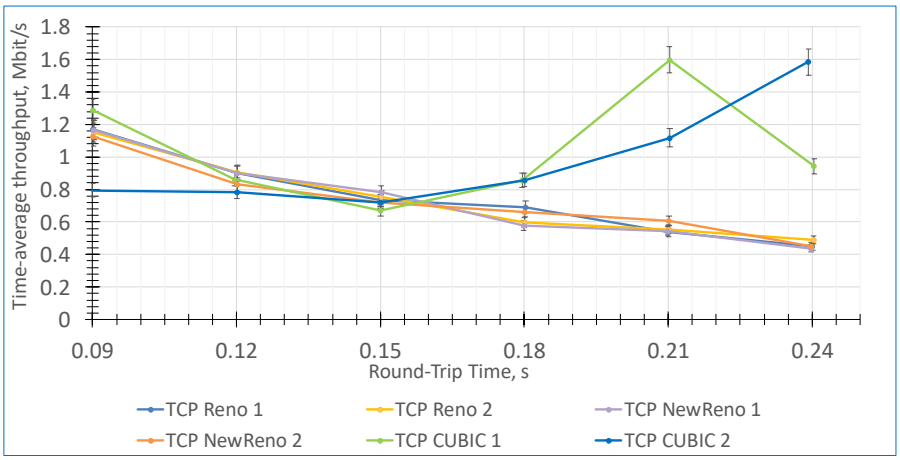


A logical scheme of a Simulation setup

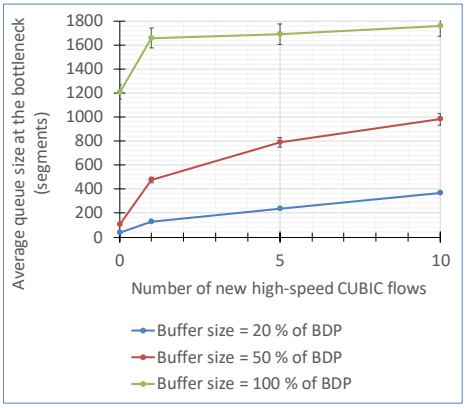
Results

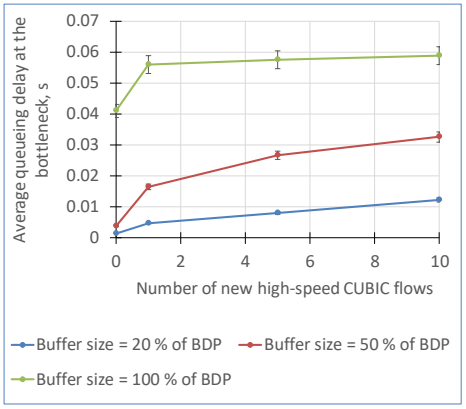
Scenario 1





Scenario 2





Discussion

- A non-linear and highly scalable window increase algorithm of CUBIC leads to a transmission of large packet bursts in the network environment with random loss, but a constant (deterministic) loss probability *p*, with *1/p* packets between two successive loss events. Fast increase of the transmission rate leads to a larger number of randomly dropped packets within a period of time.
- Loss-based approach of congestion control in CUBIC and a loss-free-time-dependent congestion window growth rate, maintain the utilization of the buffer at the maximum level.

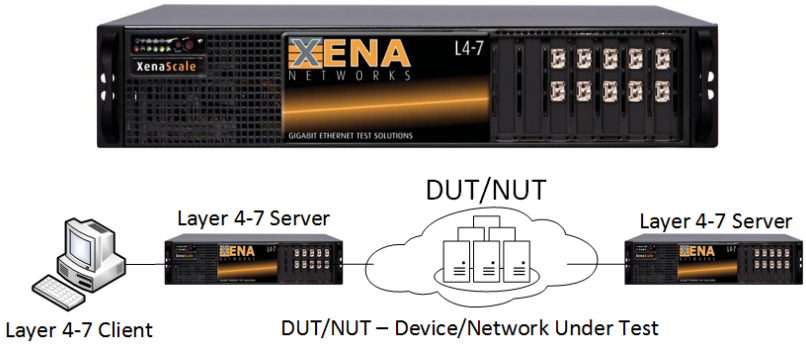
New high-speed CUBIC flows contribute to the build-up of large packet queues and increased queueing delays.

- Different buffer size settings at the bottleneck node (Router) may alleviate the performance degradation, but at the cost of decreased throughput. Hence, more advanced queue management algorithms will improve the situation to a certain extent.

Large scale network testing: challenges and opportunities

When it comes to real-life network testing as such, in order to be able to perform a large-scale network performance evaluation, there are a few available strategies, but the most realistic one is an emulation of a large number of concurrent TCP connections through a scalable testing platform. Since each established TCP connection, especially when the high-speed TCP flavours are used, maintains relatively large amount of connection state information (TCP Connection Block), significant amount of memory and computational resources might be consumed. If we consider emulation of millions of simultaneous TCP connections (a realistic scenario for a medium-sized backbone or a Data Center Network), multiple scalability problems of a network tester may arise. Thus, several alternative implementations of a TCP protocol stack have been considered, which support only the generic functionality of a TCP connection engine (requiring much less connection state information). An example of such a simplified TCP connection engine is a light-weight-IP (lwIP) stack. While it can be deployed for testing purposes, there are still several important questions to answer: will it be sufficient enough for a realistic network evaluation, and what is a scalability penalty for a high-speed TCP connection testing approach in a resource-constrained tester? These are the questions we would like to answer as a future step in our research. We have considered a L4-7 network testing platform, provided by Xena Networks ApS, as a potential environment for experimental prototyping. It offers the following testing capabilities:

- Scalable Gigabit TCP testing (1 G, 10G and 40G interfaces)
- Stateful Traffic generation (load) with 24 M TCP Clients and 24 M Servers on one platform
- Connection ramp up rate: 12 M connections/s.



Research topics of current interest

- Data Center Networks
- Cloud Computing
- SDN
- Energy Efficiency

Paper B: Robustness of multiple high speed TCP CUBIC connections under severe operating conditions

A. Pilimon, S. Ruepp and M. S. Berger, "Robustness of multiple high speed TCP CUBIC connections under severe operating conditions", in *Proc. IEEE 14th International Symposium on Network Computing and Applications (NCA)*, Boston, MA, USA, 28-30 Sept., 2015, pp. 76-80.

DOI: 10.1109/NCA.2015.43

Please note: in the following paper, the captions of two presented figures are not consistent with the presented results, namely in Fig. 7, the indicated BER value in the caption must be BER=1E-006, while in Fig. 8 this value must be BER=1E-007, respectively. Updated results with greater statistical confidence, corresponding to Fig. 6, Fig. 7 and Fig. 8 in the paper, are included next.

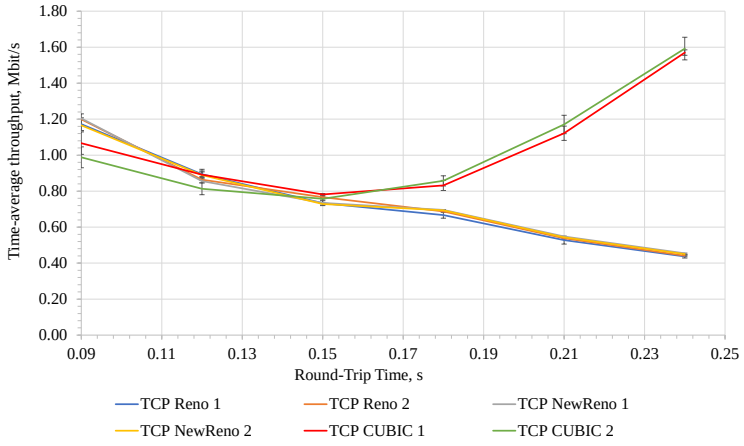


Figure 4.1: Time-average throughput as a function of BER=10^{-6} and RTT. Simulation results with 98% confidence interval after 15 simulation runs

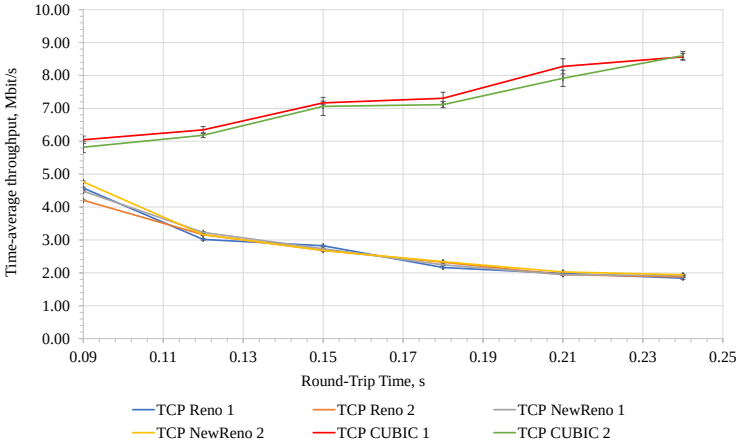


Figure 4.2: Time-average throughput as a function of BER=10^{-7} and RTT. Simulation results with 98% confidence interval after 15 simulation runs

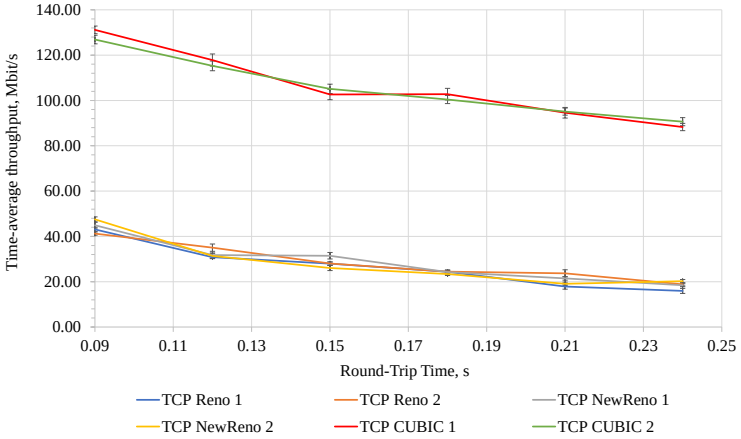


Figure 4.3: Time-average throughput as a function of BER=10^{-9} and RTT. Simulation results with 98% confidence interval after 15 simulation runs

2015 IEEE 14th International Symposium on Network Computing and Applications

Robustness of Multiple High Speed TCP CUBIC Connections Under Severe Operating Conditions

Artur Pilimon, Sarah Ruepp, Michael S. Berger
Department of Photonics Engineering
Technical University of Denmark
Kgs. Lyngby, Denmark
{artpil, srru, msbe}@fotonik.dtu.dk

***Abstract*—We study the adaptation capabilities and robustness of the high-speed TCP CUBIC algorithm. For this purpose we consider a network environment with variable and high random packet loss and a large Bandwidth-Delay product, shared by multiple heterogeneous TCP connections. The analysis is based on and supported by packet-level simulations. The results show that the aggressive nature of CUBIC's nonlinear congestion window control principle causes a degradation of the time-average throughput at the moderate level of random packet loss even under increasing Round-Trip-Time of the flow. However, this algorithmic scalability and loss-free-time-dependent window growth allows recovering transmission rate faster in the high packet loss region due to the statistically lower number of dropped packets, compared to the moderate loss level.**

Keywords—Transmission Control Protocol; congestion control; TCP CUBIC; Reno; NewReno; random packet loss; TCP fairness

I. INTRODUCTION

A major fraction of today's globally generated Internet traffic relies on TCP (Transmission Control Protocol) [1] as a transport protocol [2]. Therefore, TCP protocol plays a critical role in ensuring a connection-oriented, reliable and scalable data transport service over an unreliable IP protocol [2].

Extensive scientific research [3][4] showed that the well-known traditional TCP variants, such as TCP Reno [5] and NewReno [6] face serious performance degradation challenges, especially in terms of bandwidth utilization efficiency, on the network paths with a large BDP (Bandwidth-Delay Product) [3][4]. In order to remedy this problem, several alternative TCP variants, usually referred to as "high-speed" TCP, have been proposed and have attracted wide interest of the research community as well as the industry [3][4][7]. For example, High-Speed TCP [3], TCP CUBIC [7] or Compound TCP [8].

The abovementioned high-speed TCP flavors have been studied in theoretical-analytical and practical-experimental dimensions and are reported to offer a significant improvement in terms of higher throughput and bandwidth utilization efficiency [3][8]. However, several alternative sources [4][9][10] argued that the modern high-speed TCP variants tend to be much more aggressive in terms of bandwidth sharing fairness (get much higher bandwidth than their fair share [11]) among multiple different TCP connections, which may have to share the same network path or link. The reason is a more aggressive nonlinear congestion window increase strategy and less aggressive window back-off mechanisms of the high-speed variants [9][10], such as TCP CUBIC or High-Speed TCP.

The aim of this work was to analyze the operational stability and robustness of the high-speed TCP CUBIC connections in a network environment together with a mix of standard TCP connections (Reno, NewReno). Similar scenarios have already been investigated in [7][10][11], but our contribution is different in the following way – we considered a severe communication scenario, which allows us to reveal the operational limits of the high-speed TCP CUBIC algorithm in comparison to the standard TCP connections. We analyzed the behavior of TCP CUBIC in a high-speed and long-delay network, with initially moderate and gradually increasing random packet loss rate, and increasing RTT (Round-Trip-Time) of the flows on a 2 *Gbps* network path, shared by multiple other TCP connections. The analysis is based on and supported by the simulations using the Riverbed network modelling and simulation tool. Hence, the simulation-based analysis of TCP CUBIC was conducted in order to gather additional results and to complement an insight of the adaptation capabilities and operating robustness of multiple TCP CUBIC connections under severe operating conditions.

The reasons for investigating the operating robustness of high-speed TCP CUBIC together with TCP Reno and NewReno flavors are as follows. First, high-speed TCP CUBIC variant has a large potential of further support and improvements, because it is a current default option in GNU/Linux (since 2.6.19 kernel) and a large fraction of Internet servers are running on Linux Server [7] as a platform. Secondly, the TCP Reno and NewReno variants are analyzed together in order to have a base-case scenario to compare with, since TCP Reno/NewReno were widely accepted and deployed initially [2][4] and are still used nowadays [2]. Hence, it is very important to consider the interoperability aspects among existing and new solutions to reveal the potential problems under different operating conditions of the network, including the environments that the high-speed flavors, such as CUBIC, were not initially designed for (random packet loss patterns).

The rest of this paper is organized as follows. In Section II, we describe the simulation setup and scenario. In section III, we conduct the analysis of the obtained simulation results. Finally, the results are summarized in Section IV.

II. PACKET-LEVEL SIMULATIONS

A. Simulation setup

We consider a scenario, where multiple simultaneous TCP connections of different types (Reno, NewReno and CUBIC) are established between the Workstation (Client) and Server nodes in a network environment with dominant random packet loss pattern and increasing RTT of the TCP flows. In this scenario, we analyze the impact of the increasing random packet loss on the performance metrics (*CWND* size, throughput and inter-protocol fairness) of the mentioned TCP connections in a high-bandwidth and long-delay environment.

/15 $31.00 © 2015 IEEE
DOI 10.1109/NCA.2015.43

IEEE computer society

The logical setup of the simulated network environment is depicted in Fig. 1. This is a well-known dumbbell topology, widely used for analysis of the TCP connections and packet queueing systems. Such a network topology is sufficient enough for the purpose of our investigation, since we are focusing on the algorithmic behavior of CUBIC in a specific communication context, not including the network bottleneck and buffer sizing scenarios. The configuration settings used are as follows.

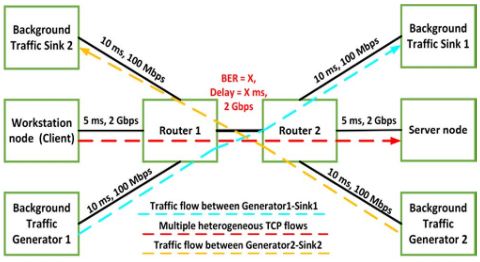


Figure 1. Logical scheme of the simulated network environment

The buffer sizes in the intermediate routing nodes (Router 1 and Router 2) are set using the BDP rule (*large buffer regime*). There are six persistent (long-lived, always have data to send) TCP connections (between Client – Server) of different types: 2 Reno, 2 NewReno and 2 high-speed CUBIC connections. These TCP flows have the same average RTT and start up at the same time. The propagation delays (one way) and the link rates are indicated in Fig. 1. The delay values are chosen based on the analysis of the available published research work in [4][7]. In addition, the bandwidth of the *transit link* (Router 1 – Router 2) is 2 *Gbps*. The initial Slow Start threshold is set to 1 Megabyte; Maximum Segment Size (MSS) is set to 1000 Bytes. Used TCP options for high-speed operation: MSS (Maximum Segment Size) and Window Scaling (WS). The Delayed ACK scheme is disabled. The receiver's buffer size and the Receive Window are scaled with the BDP of the path for each simulated RTT value. The duration of one simulation run is 1200 s (20 min.).

The bidirectional background traffic generation is used in order to add variable cross-traffic over the transit link. This includes a mix of traffic sources with exponential (random traffic) and Pareto (bursty traffic) time distributions. The rate of the background traffic is set to be a small fraction of the link's BDP, namely 2%.

B. *Varied parameters*

In this scenario, the operating conditions of the TCP flows are affected by the increasing random packet loss rate and the increasing RTT of the flows in the following sequence:

- The one-way propagation delay on the transit link is varied in steps of 15 *ms* in the following sequence from 35 *ms* to 110 *ms*: 35, 50, 65, 80, 95 and 110. Therefore, the average fixed component of the RTT of the considered flows, which traverse three links along their end-to-end path, is changing in the following sequence: 90, 120, 150, 180, 210 and 240 *ms*, accordingly.
- The random packet loss is introduced by setting the random Bit-Error-Rate (BER) with probability p on the transit link in the ascending order as follows: 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6} and 10^{-5}. It is stated in [12] that many of the lightwave systems today "specify a BER of 10^{-9} as the operating requirement". Thus, the BER level is varied from this minimum operating requirement threshold to a high level of 10^{-5}.

C. *Performance metrics*

We use the methodology for performance evaluation of the congestion control algorithms, proposed in RFC 5166 [13] and a set of the following metrics:

- The evolution of the average *CWND* size (in MSS-sized segments).
- The evolution of the time-average throughput (*Mbit/s*).
- The inter-protocol fairness of bandwidth sharing between multiple different TCP flavors with the same average RTT of the flows. A Jain's Fairness Index [13] is used for this evaluation. It is calculated using (1):

$$f(x_1, x_2, x_{3,\dots,} x_N) = (\sum_{i=1}^{N} x_i)^2 \,/\, (N \cdot \sum_{i=1}^{N} x_i^2) \qquad (1)$$

In (1), x_1, x_2, $x_3 \dots x_n$ are the throughputs of the TCP flows, N is the total number of TCP flows, traversing the path/link.

III. RESULTS

A. *CWND and throughput evolution in TCP CUBIC*

As it can be seen in Fig. 2, the average CWND size (the results are provided only for one connection for clarity) grows with the RTT of the CUBIC flow as expected (approximated by the trend lines), because the duration of the congestion-free epoch defines the size of the window increment and, as a result, the CWND growth rate. This behavior is in agreement with the analytical expression (2), proposed by the authors of CUBIC [7] for the estimation of the average CWND size.

$$E\{CWND_{CUBIC}\} = 1.17 \cdot \sqrt[4]{(RTT \,/\, p)^3}, \qquad (2)$$

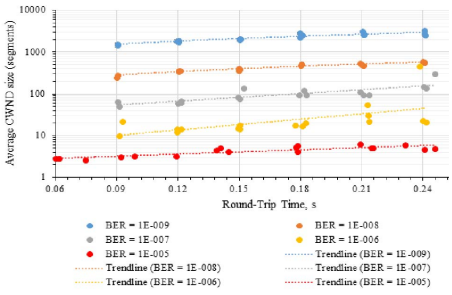


Figure 2. The average CWND size as a function of BER and RTT for TCP CUBIC (connection 5). Simulation results with 3 seed values per run

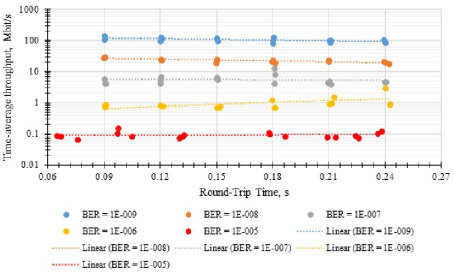


Figure 3. Time-average throughput (in Mbit/s) as a function of BER and RTT in TCP CUBIC (connection 5). Simulation results with 3 seed values per run

Considering the impact of the random packet loss pattern on the connection performance under high BDP conditions, it can be noticed in Fig. 3, that the increasing BER on the transit link dramatically degrades the time-average throughput of a high-speed CUBIC connection. The throughput drops by, approximately, a factor of five, when the BER level increases by a factor of ten. This trend is valid until the BER level rises up to 10^{-5}, corresponding to an average packet loss of 7.89% (value was calculated using the relation of the packet loss probability, packet size and the BER on the link assuming no error correction).

When it comes to the throughput degradation, mentioned before, we can notice (see Figures 2 and 3) that even though the average CWND size is gradually increasing with the growth of RTT, the time-average throughput is somewhat decreasing in the moderate loss region (1E-009, 1E-008) and remains constant or tends to increase in the high loss region (1E-007, 1E-006). The reason is that the CWND growth rate in CUBIC is dependent on the duration of the loss-free period and increases with the RTT. Since the window grows fast due to the cubic function and becomes relatively large in a short time, this burst of data packets is sent to the network environment with random loss, but a "deterministic" loss probability p with $1 / p$ number of packets (on average) between two successive loss events [3][7]. Thus, the larger the number of packets sent at a time, the more loss events will occur within a period of time and the larger number of Fast Recovery/Fast Retransmit actions will be performed. In addition, Fast Convergence mechanism will additionally reduce the maximum CWND size as explained in [7].

There is one more important aspect to consider, resulting from large BER levels, namely the RTT measurements of the TCP flow are distorted, because multiple closely spaced packets may be dropped and, if the *Karn's* RTT estimation algorithm is enabled (default option), the RTT samples are not taken for the retransmitted packets. As a result, gaps in RTT sampling may occur and the calculated average smoothed RTT value as well as measured RTT variance might be very inaccurate (see Figures 2, 3, 5 and 9 under BER = 10^{-5}). Another reason is the insufficient number of RTT samples due to too small CWND size, leading to an average RTT with a relatively large error (when the Timestamps are not used) and an imprecise RTO timer. The latter may result in a burst of spurious retransmissions (consequence of a more aggressive window growth in CUBIC) due to the timeout, degrading the overall performance even more.

B. *CWND and throughput evolution in TCP Reno/NewReno*

The performance metrics for each of the standard TCP connections (Reno and NewReno) were obtained via simulations in a similar way as in part *A*. The results show (see Fig. 4) that the average CWND size of the Standard TCP is inversely proportional to the square root of the packet loss rate, meaning that the window decreases with the increase of the BER level as expected [3] and these results are widely known. In addition, there is no visible difference between the CWND growth pattern in Reno and NewReno, since they use almost the same general principles of congestion control and NewReno may only outperform Reno in case of multiple losses within the same window of data (e.g. synchronized loss pattern due to buffer overflow).

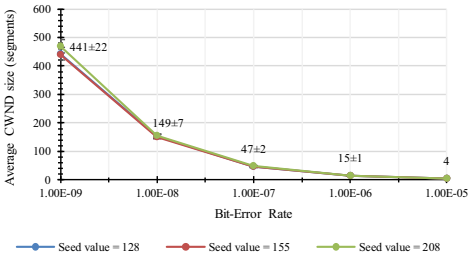


Figure 4. The evolution of the average CWND size as a function of BER in Reno (connection 1). Simulation results with 3 seed values and 95 % confidence interval

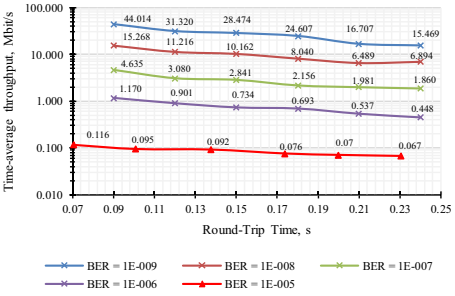


Figure 5. Time-average throughput (in Mbit/s) as a function of BER and RTT in Reno (connection 1). Simulation results, averaged for 3 seed values.

The average throughput of a standard Reno/NewReno connection decreases when the average experienced delay is increasing (see Fig. 5 for a Reno connection). The reason is very well known and is rooted in a very *conservative window increase by a maximum of 1 MSS-sized segment per RTT*. Therefore, when the RTT is increasing, there are fewer window updates per time unit and the throughput drops.

C. *Throughput evolution among multiple mixed TCP Reno, NewReno and CUBIC connections*

Recalling that the entire communication path has 2 Gbps of available bandwidth in this simulation scenario, Fig. 6 presents the aggregate time-average throughput of all 6 simulated TCP connections and shows the consequences of the impact of the increasing BER on the transit link. While CUBIC is known to be very efficient in the wired environments with large BDP and low non-random loss rates [7], it can be seen that even at a moderate BER level of 10^{-9}, both CUBIC flows manage to utilize on average only around 6 – 7% of the available bandwidth each, reaching about 130 Mbps at 90 *ms* RTT. The standard TCP connections, in turns, suffer from the aforementioned underutilization problem and can efficiently use only around 2.2% of the bandwidth each, resulting in no more than 45 Mbps per flow at 90 *ms* RTT.

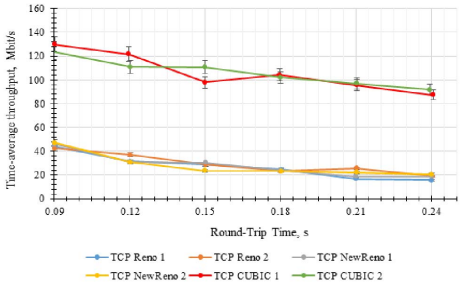


Figure 6. Time-average throughput (in Mbit/s) as a function of BER = 1E-009 and RTT. Simulation results with 95 % confidence interval

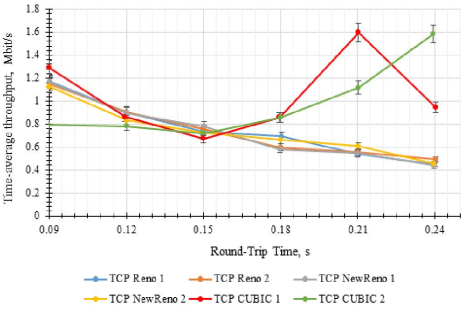


Figure 7. Time-average throughput (in Mbit/s) as a function of BER = 1E-007 and RTT. Simulation results with 95 % confidence interval

However, as it can be seen in Fig. 7 and Fig. 8, when the BER level increases significantly and CUBIC is operating in the "high" loss region (1E-007, 1E-006), the time average throughput tends to increase in a more stochastic way with the increase of RTT. In contrast with the previous situation, where the efficiency of CUBIC was limited by its own window growth strategy, in this case the situation is different. Since the average CWND size is very small (at the order of several tens of packets) and comparable to the standard TCP, there is no large burst of data, and fewer packet loss events are occurring now, because we have a "deterministic" loss probability. As a result, CUBIC is able to utilize its congestion-free-time-dependent window growth strategy and cubic function allows to catch up with the window size faster. The main reason for such "fluctuations" seen in the time-average throughput may be insufficient number of seed values in the simulations, affecting the accuracy of the results and leading to some stochastic variations, and this needs to be addressed.

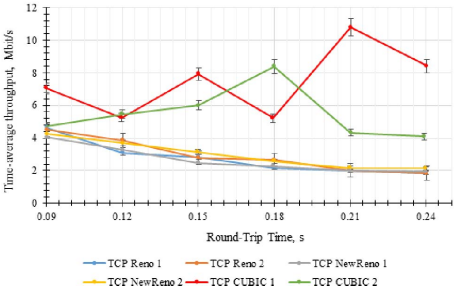


Figure 8. Time-average throughput (in Mbit/s) as a function of BER = 1E-006 and RTT. Simulation results with 95 % confidence interval

It is important to emphasize, why the number of TCP connections in our analysis was limited. First, if the chosen number of concurrent TCP connections, high speed TCP CUBIC in particular, would be large, our investigation would converge to the analysis of two different loss patterns mixed together, namely the random packet loss pattern (our primary focus here) and the synchronized pattern due to the buffer overflows in one of the routers. The reason is that a sufficiently large number of CUBIC flows would take over all the available bandwidth very fast and would saturate the router's buffer leading to bursts of packet losses. In that case, the performance analysis would deflect from our initial goal and it would be difficult to separate the impact of random losses from the synchronized losses. This is why the number of connections was limited to avoid introducing extra losses and confusion in the analysis.

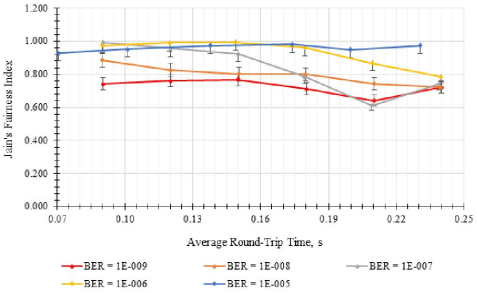


Figure 9. Bandwidth sharing fairness among 6 TCP connections as a function of BER and the average RTT. Simulation results (averaged for 3 seed values)

The simulation results of the bandwidth sharing fairness among different TCP connections are depicted in Fig. 9. Jain's Fairness Index indicates whether the available capacity of the path (or link) is partitioned in locally equal proportions between the existing flows, which traverse the same path or link, or not [13]. The lower the value of this index (bounded to a maximum of one), the larger the bandwidth usage unfairness among the flows (high-speed flows with respect to the others).

In our case, Jain's Index shows that the bandwidth sharing is much less fair when the BER level is moderate (1E-009, 1E-008), and becomes more fair, when the BER is increasing. The reasons for that were explained before in terms of the CWND growth specifics for CUBIC and standard TCP. Under the moderate packet loss, the high-speed CUBIC flows utilize their more aggressive window increase mechanism and, on average, get more bandwidth than the other regular flows. The unfairness becomes more severe (the index decreases) with the increase of the average RTT. The reason is as follows: it can be seen in Fig. 6, that the standard TCP flows suffer from the RTT-dependency of the window growth more than CUBIC. This is because in the presence of the moderate loss rate, the latter still maintains its throughput relatively high (within a range of RTTs, the throughput drops from ~ 130 to around 90 Mbps) due to the aggressive, but flexible window increase strategy. On the other hand, the standard TCP flows lose more than half of their throughput within the same range of RTTs (from ~ 45 to 20 Mbps). In the "high" loss region (1E-007, 1E-006), sharing remains fair due to the efficiency problems (switching between TCP-friendly [7] and scalable mode) of CUBIC, and the Index is the highest, when all the connections suffer from degraded performance in the highest loss region (1E-005), where CUBIC operates in the TCP-friendly mode. The TCP-friendly mode of operation helps to maintain the efficiency at least the same as in the standard TCP.

IV. CONCLUSION

In this work we conduct a simulation-based analysis of the robustness of high-speed TCP CUBIC connections under increasing random packet loss in a network environment with large Bandwidth-Delay product and different TCP connections.

The results show that the aggressive nature of the window increase algorithm of TCP CUBIC leads to a degradation of the time-average throughput even under the increasing Round-Trip Time of the flow and a moderate level (the BER of 10^{-9} - 10^{-8}) of random packet loss. However, a different effect was observed in the "high" packet loss region (corresponding to BER of 10^{-7} - 10^{-6}), where the scalable cubic window growth function allowed CUBIC connections to recover the transmission rate faster with the increase of the average Round-Trip Time, compared to the moderate loss region. The reason for such a behavior of the CUBIC flows is the statistical dependency of the number of randomly dropped packets on the current size of the congestion window, meaning that high-speed flows with large windows, operating under large Bandwidth-Delay product conditions, are affected more due to larger bursts of packets sent to the network within the transmission round (leading to more frequent packet losses and Fast Retransmit/Recovery actions). On the contrary, small (several tens of packets) size of the congestion window in the high loss region results in a small burst of packets and statistically fewer dropped packets within a transmission round. Some stochastic "artifacts" in the results are most likely caused by the small number of simulation seeds and will be addressed.

ACKNOWLEDGMENT

This work has been partially supported by the Danish Innovation Foundation through the project "Layer 4-7 testing at 100 Gbps".

REFERENCES

[1] J. Postel, *Transmission Control Protocol,* RFC 793, IETF, 1981.

[2] A. S. Tanenbaum and D. J. Wetherall, "The Internet Transport Protocols: TCP," in *Computer Networks*,, Fifth ed., Prentice Hall, 2011, pp. 552-581.

[3] S. Floyd, *HighSpeed TCP for Large Congestion Windows,* RFC 3649, Experimental, IETF, 2003.

[4] Y.-T. Li, D. Leith and R. N. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks," *IEEE/ACM Transactions on Networking,* vol. 15, no. 5, pp. 1109-1122, October 2007.

[5] M. Allman, V. Paxson and E. Blanton, *TCP Congestion Control (Reno),* RFC 5681, IETF, 2009 .

[6] T. Henderson, S. Floyd, A. Gurtov and Y. Nishida, *The NewReno Modification to TCP's Fast Recovery Algorithm,* RFC 6582, IETF, 2012.

[7] S. Ha, I. Rhee and L. Xu, "CUBIC: A New TCP-friendly High-Speed TCP Variant," *ACM SIGOPS Operating System Review*, vol. 42, no. 5, pp. 64-74, 2008.

[8] K. Tan, J. Song, Q. Zhang and M. Sridharan, "A Compound TCP Approach for High-Speed and Long Distance Networks," in *Proceedings of IEEE INFOCOM,* 2006.

[9] J. Chicco, D. Collange and A. Blanc, "Simulation Study of New TCP Variants," in *The IEEE symposium on Computers and Comm.*, 2010.

[10] S. Poojary and V. Sharma, "Theoretical Analysis of High-Speed Multiple TCP Connections through Multiple Routers," in *2013 IEEE International Conference on Communications (ICC)*, 2013.

[11] R. Oura and S. Yamaguchi, "Fairness Comparisons Among Modern TCP Implementations," in *26th International Conference on Advanced Information Networking and Applications Workshops*, 2012.

[12] P. G. Agrawal, Fiber-Optic communication systems, Fourth ed., A John Wiley & Sons, Inc., 2010.

[13] S. Floyd, *Metrics for the Evaluation of Congestion Control Mechanisms,* RFC 5166, Informational, IETF, 2008.

Paper C: Energy efficiency benefits of introducing optical switching in Data Center Networks

A. Pilimon, A. Zeimpeki, A. M. Fagertun and S. Ruepp, "Energy efficiency benefits of introducing optical switching in Data Center Networks", in *Proc. of Workshop on Computing, Networking and Communications (CNC)*, Santa Clara, CA, USA, 26-29 Jan., 2017, pp. 891-895.

DOI: 10.1109/ICCNC.2017.7876250

Energy Efficiency Benefits of Introducing Optical Switching in Data Center Networks

Artur Pilimon, Alexandra Zeimpeki, Anna Manolova Fagertun and Sarah Ruepp
Department of Photonics Engineering
Technical University of Denmark
Kgs. Lyngby 2800, Denmark
{artpil, s141834, anva, srru}@fotonik.dtu.dk

***Abstract*—In this paper we analyze the impact of WDM-enhanced optical circuit switching on the power consumption of multiple Data Center Network (DCN) architectures. Traditional three-tier Tree, Fat-Tree and a ring-based structure are evaluated and optical switching is selectively introduced on different layers of the network topology. The analysis is based on network-level simulations using a transport network planning tool applied to small-scale setups of the considered DCNs. The obtained results show that introducing all-optical switching within the DCN leads to reduced power consumption in all architectures, except the traditional Tree. This is caused by the inability to perform efficient traffic grooming and smaller average nodal degree of this architecture. Enabling optical switching only at the aggregation layer results in the highest energy savings in Fat-Tree and traditional Tree, while an optically switched core benefits most the ring-based network. For the latter, the core ring nodes need fewer long-reach transponders at the trunk interfaces and benefit from more efficient traffic grooming in the access part.**

Keywords—Data Center Network; energy efficiency; optical switching; Optical Cross Connect; opaque network

I. INTRODUCTION

Almost every single aspect of our daily life is directly or indirectly related to some form of data processing, ranging from sending an e-mail, watching a video on YouTube, using Google Search or social networking and messaging services (Facebook, Twitter, etc.) to more complex financial operations, scientific data mining and analytics or cloud-based services just to name a few [1]. Most of these communication services may require timely interaction and data exchange between multiple functional components, before the final response message or a requested service is delivered to the end user. This means that the applications and services define the communication and processing needs, and in this way pose certain requirements on the compute, storage and network infrastructure of a Data Center (DC) [2].

Despite the fact that the DCs and the available DCN infrastructure have become the backbone of the global economy, being as important as the road infrastructure or the Internet as such, these facilities are becoming more and more power demanding components of the global ICT infrastructure, contributing to an increasing fraction of the overall carbon footprint (Green House Gases emissions) and power usage [3] [4] [5] [6]. According to [3], the DCs contributed to the global ICT GHG emissions by 14% (2007), with an expected growth of up to 18% by 2020; as of 2010, these emissions were equivalent to the GHG emissions produced by 70 – 90 large (around 500 MW) coal-fired power plants [5]. The amount of energy consumed by the DC facilities was estimated to be 1.5% of the total energy consumed within the US in 2006 [3] and reached 91bn kWh in 2013 (with a projected increase by 47bn kWh by 2020 that is equivalent to 13bn USD annually), while the global DC energy demands were estimated to be 330bn kWh (2007), and this parameter is predicted to exceed 1000bn kWh by 2020 [3]. Therefore, energy efficiency is becoming an increasingly important factor, defining the future of the communication technologies to be used.

It is important to point out the main reasons and factors contributing to such an enormous power usage in the DCs. First, there have been unprecedented growth of the global data traffic volumes due to the deployment of new services (social networking, multimedia streaming, cloud computing services), resulting in a need for expansion of the network infrastructures and a large number of new DCs being built all over the world [1] [2]. Second, the IT and networking equipment consume between 40 and 92% of the total DC power supplied, depending on the architectural solutions used. This energy is shared by the servers (~35 – 40%), storage devices (~30%) and network devices (20 – 25%) [2] [4], while the remaining power is consumed by the supporting infrastructure (power supplies, air conditioning and cooling, etc.) [7]. Third, the nature of the traffic generated within the DCs is such that most of the traffic (~75 %) remains within the DC (traffic locality) where most of the communication between different components of an application or service takes place (multi-tier applications) [1]. Moreover, a large number of deployed DCNs use power demanding Optical-Electrical-Optical (O-E-O) signal conversion [3]. As a cumulative result of these factors, the number of servers will tend to increase significantly [1]. Therefore, serious concerns are being raised about the cost effectiveness and energy efficiency of DC facilities [2] [4].

Advances in the field of optical communications have moved communication networks to a new level, and the idea of introducing optics in the DCN environment has commenced by using optical point-to-point links in DCNs [2]. There have been multiple research initiatives targeting performance, scalability and energy efficiency of the data plane as well as flexibility (programmability) of the control plane by combining the optical technologies with Software-Defined Networking (SDN)

978-1-5090-4588-4/17/$31.00 ©2017 IEEE

solutions, respectively [2] [8]. In addition, potentials of using photonic integrated circuits (PICs) and new silicon photonics devices as well as multi-core fibres (MCF), Space and Wavelength Division Multiplexing (SDM and WDM, accordingly) combined with software-level solutions (energy-aware routing, VM migration, resource scheduling, as well as Network Function Virtualization) [2] [5] are projecting long-term benefits for DCNs [2] [8]. (The impact of the aforementioned improvements is not considered in this paper.) Nevertheless, the commonly used O-E-O conversion at every network node, packet buffering in every Electronic Packet Switch (EPS) and electrical packet switching significantly contribute to the overall intra-DC delay and energy consumption and will tend to increase with the network scale.

The goal of this work is to investigate the feasibility of applying WDM-enhanced optical switching at different layers of the topology (access/edge, aggregation, core or a combination thereof) or considering all-optical switching within the entire DCN interconnect as compared to a baseline case with no optical switching at all, in the context of power consumption in different considered DCN architectures, namely the traditional Clos-based three-tier Tree [9], Fat-Tree [10] and a Ring-based structure [11]. This is an indicative study, which should be treated as exploration of opportunities, offered by optical switching technologies, and the results of this work can be used to form a better insight on where in particular (which layer(s)) it would be beneficial to introduce optical switching. We would like to emphasize that we perform network dimensioning (targeting optical switching based on static all-to-all communication demands), and do not consider optical switching based in dynamic traffic patterns.

The remainder of this paper is organized as follows: In Section II, we provide an overview of the state-of-the-art in this area. In Section III, we describe the methodology, our idea and simulation setup, used components and considered scenarios. Section IV presents the obtained results and Section V concludes the paper.

II. State-of-the-Art in optics for Data Centers

There has been a lot of focus in the research community on alternative architectural designs (including custom hardware and software), new silicon photonic materials and structures (devices), aiming to optimize the cost, performance parameters (latency, bandwidth, energy efficiency) as well as the environmental impact of DCNs. Several hybrid DCN architectures have been proposed for the network interconnect part of a DC, incorporating electronic and/or optical switching devices. These architectural solutions are discussed as follows.

Kitayama in [12] proposed a high-performance optoelectronic intra-DC packet switching (based on Optical Packet Switching (OPS)) network, using Hybrid Optoelectronic Routers (HOERs), interconnected with 100 Gb/s optical links (4x25 Gb/s). This work was substantially enhanced in [13] by: a) a combination of OPS and Optical Circuit Switching (OCS); b) DC network structure being converted to N-Dimensional Torus topology. All these enhancements are reported to bring significant improvement in energy efficiency, namely 0.09 W/Gb/s (extracted from 120 W/1280 Gb/s) compared to 2.25 W/Gb/s (360 W/160 Gb/s) in the previous version of HOER [13]. However, in the evaluation of energy efficiency the authors assess only two scenarios, namely fully electrical and all-optical network configurations (Fat-Tree and 6-D Torus). In addition, they use hypothetical power consumption data for the optical core switch (0.3 W/Gb/s), which they aim to achieve.

The authors of [14] present a complex all-optical DCN architecture called Optical Pyramid DC (OPMDC) network, composed of three types of WSS-based (Wavelength Selective Switch) nodes on 3 network levels (tiers). In addition to being very scalable, modular, low-latency and offering ultra-high throughput, this architecture is reported to be five times more power efficient than DCNs with E-switches of similar configuration [14]. For example, in a structure with 343 48-port OvS (Optical virtual Switch)/ROADM nodes (i.e., 16 464 ports in total), the typical power consumption is around 1 W per port [14] on tier-1. Its operational efficiency is enhanced by using SDN-based control architecture. However, the experimental prototyping was limited to using customized and vendor-specific OvS/ROADM nodes. Hence, energy efficiency metrics may need to be evaluated more objectively.

In [15], the authors presented a HOSA (Hybrid Optical Switch Architecture) structure for DCN interconnect, realized as a two-level topology with a single stage of all-optical core switches (OCS) and electrical layer of ToR switches. The core is composed of slow MEMS-based (Micro-Electromechanical System) optical switches and fast AWGRs (Arrayed Waveguide Grating Routers) or SOAs (Semiconductor Optical Amplifiers) in the core layer, providing a trade-off between cost, energy efficiency and switching speed of the interconnect. Simulation studies show that this architecture brings 20-30% improvement in power consumption as compared to the Fat-Tree, traditional electrical (TE) or hybrid optical-electrical (OE) switches, but is more expensive to deploy. An enhancement of this architecture was presented in [16], where instead of using OCS or OPS, authors use OBS (Optical Burst Switching) with a two-way reservation protocol, and use a centralized optical control plane. A detailed power consumption analysis shows that this solution achieves 65-70% improvement in power consumption over BCube and Fat-Tree networks, as well as 27-33% improvement over TE and OE interconnects under the same configurations (around 40k servers).

More recent work on the next-generation DCN architectures has been done as a part of an ongoing European research project, called COSIGN (Combining Optics and SDN In next Generation data centre Networks) [8]. The main goal of this project is to identify, develop and implement a flat and highly scalable DCN architecture enhanced by a custom SDN-based control and service orchestration platform. Two main long-term DCN architectures are proposed, namely a meshed structure and a ring-based structure [8]. In the former, the key introduced features and components are: optical WDM ToR (Top-of-Rack) switches, optical server NICs (Network Interface Cards), a combination of several types of inter-rack connectivity based on fast 4x4 optical switches using OTDM, OPS and OBS (Optical Burst Switching), and an option to use large scale fibre switches with MCF and SDM [8]. An example of the latter, namely a ring structure, called Ring of Rings

(RoR) for all-optical DCNs with reported improvement (by 40-99%) in connection request blocking as well as 3-17% improvement in resource utilization, is presented in [11]. Both architectures are projected to significantly reduce power consumption of the data plane [8].

In [17], the authors present a project called PhoxTroT, focusing on the development of optical chips and small-size boards for several hierarchical levels of DCN and HPC (High Performance Computing) environments, namely on-board, board-to-board and rack-to-rack optical interconnect components. The main features of these systems are low-power and cost, small size and high performance; e.g., an Active Optical Cable (AOC) with an aggregate rate of 1.28 Tb/s and lower than 5mW/Gb/sec power consumption [17]. Chen *et al.* in [18] propose an optical DCN architecture with high performance, scalability and fault-tolerance being the main features, achieving up to ~65% reduction in power consumption compared to the Fat-Tree and c-Through [18].

A WDM-based approach has been considered as a promising candidate for building highly scalable and efficient future DCN interconnects, and recently this idea has started gaining the momentum [2] [8] [19] [20].

To the best of our knowledge, none of the discussed works have attempted to investigate the impact of per-network-layer (separate and/or combined) application of optical switching, enhanced by the WDM capabilities. Therefore, in this work we evaluate such an approach and take the discussion of a DCN power consumption problem to a new level by exploring the potentials of adding a DCN-oriented and WDM-enhanced optical circuit switching method using an optical network dimensioning approach based on the static uniform traffic demands and available power consumption data from industry.

III. Methodology and Simulation Setup

We consider an opaque network at the optical channel layer, where optical communication channels are being reconstructed (regenerated) at every intermediate network node of the considered DCN architectures. The main idea explored in this work is whether it is potentially feasible (in terms of energy efficiency) to deploy optical circuit switching, selectively applied at particular layers of the considered DCN architectures and enhanced by the application of the Wavelength Division Multiplexing (WDM).

The optical network modelling tool SP Guru Transport Planner [21] was used for network dimensioning with the following settings: link-by-link traffic grooming, diverse routing (representing the Equal-Cost Multi-Path (ECMP) routing), and no protection. Additional constraints introduced: we use a combined metric to set the cost of the paths to the same value for the multi-path routing and to choose the nearest nodes as preferred). Tree types of network nodes are deployed during the dimensioning process, namely an ECC (Electrical Cross Connect), Electrical/Optical Cross-Connect (EOCC) and an Optical Cross Connect (OCC) node. A high-level structure of ECC and EOCC is depicted in Fig. 1 (shown in one Figure for space saving purpose), while the OCC is presented in Fig. 2. As it can be seen in Fig.1, the EOCC node consists of a Digital Cross Connect (DXC) and an Optical Cross Connect (OXC). In the context of a DCN, DXC can be a network device with electrical switching functionality (e.g., an Ethernet or InfiniBand switch) operating at the Digital Client Layer (DCL) and being part of a logical network topology, while OXC operates at the Optical Channel (OCH) layer of the network and performs optical channel (wavelength) switching. Short-reach (SR) transponders provide an interface for the signals (Ethernet, SDH/SONET, etc.) from the client devices (DCL layer), whereas long-reach (LR) transponders are able to convert (map) the standard client optical signal of 1310 *nm* ("gray" interface) to a WDM-specific wavelength in the 1500 *nm* region (colored interface) and provide an interface to a WDM multiplexer.

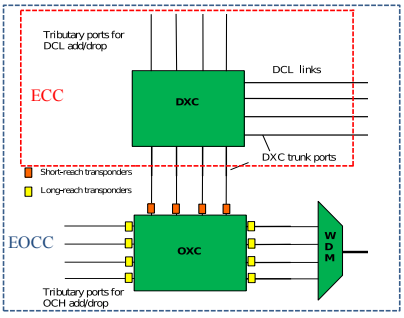


Fig. 1. Electrical (ECC) and Electrical/Optical Cross-Connect (EOCC) nodes

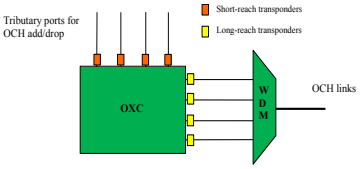


Fig. 2. Optical Cross-Connect (OCC) node

TABLE I CONSIDERED TOPOLOGIES AND THEIR PROPERTIES

Topology	Summarized properties			
	Number of network nodes, N	***Number of Links (bidir), L***	***Avr. Nodal degree,*** $\overline{N}_{deg}$	***Avg. mesh degree,*** $\overline{M}_{deg}=\overline{N}_{deg}/(N-1)$
Traditional Tree	14	48	3.43	0.26
Fat-Tree	20	80	4	0.21
Ring-based structure	12	48	4	0.36

The third node (Fig. 2) contains only an OXC and switches only optical channels at the OCH layer. With regard to the network topologies considered in our study, traditional Tree, the Fat-Tree and the ring structure, depicted in Fig. 3 – 5, respectively, the network nodes can be ECC, EOCC, OCC or several types, depending on the configuration of any particular layer. Table I contains a summary of the properties of the considered network topologies, including the average nodal

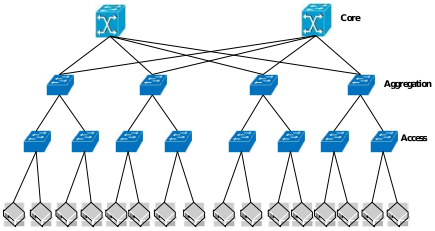


Fig. 3. A traditional multi-tier Tree DCN

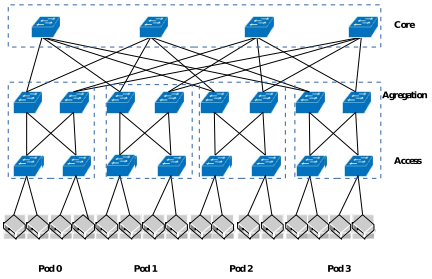


Fig. 4. A Fat-Tree (folded Clos) DCN

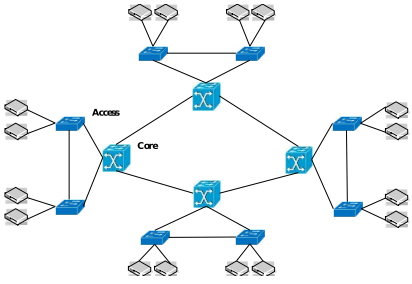


Fig. 5. A ring-based DCN

degree and a mesh degree (defined as a ratio of the average nodal degree of this topology to a nodal degree of mull-mesh network with the same number of nodes).

The following scenarios are considered: (a) *For the traditional and Fat-Tree topologies*: no optical switching, all-optical switching, only optical core switching, only optical aggregation switching, only optical access switching and optical core and aggregation layer switching; (b) *For the ring-based topology*: no optical switching, all-optical switching and optical core switching.

TABLE II POWER CONSUMPTION REFERENCE VALUES

	Average Power consumption (W/port) per device type				
Port rate (Gb/s)	SR transp.	LR transp.	Electrical (DXC) node	Electrical-optical (EOCC) node	Optical (OXC) node
1	0.3	0.6	2.7	< 2 W/Gb/s	0.3
40/OTU3	1.4	3.4	15	< 2 W/Gb/s	3.5
100/OTU4	3.4	3.4	24	< 2 W/Gb/s	3.5

These networks were dimensioned to support the uniform traffic demands between 16 server nodes, communicating in an all-to-all fashion at 1 Gb/s full capacity for every pair of servers. The power consumption of the nodes is calculated using the specification data (extracted from [22] [23] [24] [25] [26]) summarized in Table II. These values indicate an average maximum power per port (which is port rate dependent) under full load. The power calculation includes the total power including the power used by the internal components of the deployed nodes (circuitry, memory, line cards, transceivers, switch fabric). We derive the power required per port from the available data per node considering the interface types (rates), unless the direct per-port value is provided by the module vendor, for all the indicated devices.

IV. RESULTS

The results are presented in Fig. 6. The power consumption of a traditional multi-tier Tree with no optical switching capabilities is used as a reference baseline scenario. We observe that without optical switching, the power consumption of the Fat-Tree and ring architectures is significantly higher than in the traditional tree. This is due to the fact that Fat-Tree is built as a folded Clos topology with a higher degree of multi-path connectivity (number of nodes and links, especially the redundant intra-pod connections), offering full bisection bandwidth. Thus a larger volume of traffic is being spread over and may be processed within the pod, rather than going up to the core layer. This leads to larger concentration of traffic between the access-aggregation layers and larger number of transceivers and ports need to be deployed in the pod switches (48 deployed OCH links and 96/96 utilized wavelengths). In

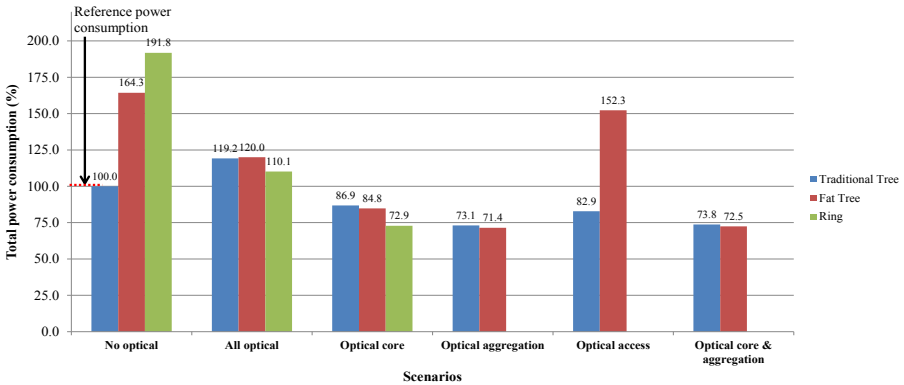


Fig. 6. Simulation results. Relative power consumption of different considered DCN architectures with selectively applied WDM-enhanced optical switching at particular network layers. Power consumption of a traditional multi-tier Tree with no optical switching is used as a baseline.

the ring there is a significantly larger concentration of traffic within the core due to a large degree of aggregation and limited multi-path routing options. This results in a large number of ports and transceivers to deploy (tributary OXC and trunk DXC ports), but in this small-scale setup the ring network has a large *nodal degree* and largest *mesh degree* (level of connectivity to the peers, see Table I) and may benefit from shorter paths than in the Fat-Tree, where the mesh degree is lower due to much larger number of nodes deployed.

The all-optical scenario reduces power consumption in the Fat-Tree and ring topologies due an opportunity to eliminate the DXCs and reduce the number of trunk ports and LR transponders at OXCs' trunk ports. This is of importance, considering that Fat-Tree and ring have larger nodal degree than a simple Tree. In contrast, power consumption of a regular Tree is slightly higher due to inefficient traffic grooming at the access layer in particular (32 OCH links established).

Optical switching in the core and/or aggregation layers (separately or combined) provides highest power savings because the traffic is more efficiently groomed at the now electrical-optical access layer. The core and aggregation layers of all three architectures are offloaded by optical switching bypassing the power demanding electrical DXCs and associated transponders. As a result, the optical core is the most beneficial on the ring network. This is because the core ring nodes need fewer long-reach transponders at the trunk interfaces, as well as intensive grooming at the access part and optical switching along the heavily aggregated ring links. We can also observe the detrimental effect of introducing access-only optical switching in the Fat-Tree. The reason is lack of traffic grooming in the access part, reflected in the increased wavelength demands at the access layer, in addition to extra DXC ports as well as SR and LR transponders.

V. Conclusions

The energy efficiency of several DCN architectures, namely the traditional Tree, the Fat-Tree and a ring-based network, has been analyzed in the context of WDM-enhanced optical circuit switching, selectively applied on particular layers of the topology. The results show that optical switching should be deployed after a thorough consideration of the network architecture, traffic distribution within the network, as well as the availability of the resources (wavelengths). In general, access-only optical switching should be avoided due to inefficient traffic grooming conditions (no grooming is possible), resulting in high wavelength demands. Optical aggregation and/or core should be considered for all-optical switching implementation on these layers.

References

[1] Cisco Systems, Inc., "Cisco Global Cloud Index: Forecast and Methodology, 2014–2019," Cisco, 2015.

[2] H. Liu, R. Urata and A. Vahdat, "Optical Interconnects for Scale-Out Data Centers," in Optical Interconnects for Future Data Center Networks, New York, Springer Science, 2013, pp. 17 - 29.

[3] C. Kachris, K. Bergman and I. Tomkos, "Introduction to Optical Interconnects in Data Centers," in Optical Interconnects for Future Data Center Networks, New York, Springer, 2013, pp. 3-13.

[4] C. Gough, I. Steiner and W. A. Saunders, Energy Efficient Servers: Blueprints for Data Center optimization, APress Open, 2013.

[5] Natural Resources Defense Council (NRDC), "Data Center Efficiency Assessment," NRDC, New York, 2014.

[6] Green Data Net Project, "Working to reduce the environmental impact of the data explosion," 2016. [Online].

[7] Y. Ohsita and M. Murata, "Optical Data Center Networks: Architecture, Performance, and Energy Efficiency," in Handbook on Data Centers, NewYork, Springer, 2015, pp. 351-390.

[8] J. I. Aznar *et al*., "COSIGN: Combining Optics and SDN In next Generation data centre Networks. Deliverable D1.4 (Public). Architecture design," CORDIS, 2016.

[9] Y. Liu, J. K. Muppala, M. Veeraraghavan, D. Lin and M. Hamdi, Data Center Networks: Topologies, Architectures and Fault-Tolerance Characteristics, Springer, 2013.

[10] M. Al-Fares, A. Loukissas and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *in Proc*. of ACM SIGCOMM conference on Data communication, Seattle, 2008.

[11] A. M. Fagertun, M. Berger, S. Ruepp, V. Kamchevska, Galili, M. Galili, L. K. Oxenløwe and L. Dittmann, "Ring-based All-Optical Datacenter Networks," *in Proc*. of ECOC, Valencia, 2015.

[12] K. Kitayama, S. Debnath, Y. Yoshida, R. Takahashi and A. Hiramatsu, "Energy-Efficient, High-Performance Optoelectronic Packet Switching for Intra-Data Center Network," *in Proc.* of ECTON, Cartagena, 2013.

[13] K. Kitayama *et al*., "Torus-Topology Data Center Network Based on Optical Packet/Agile Circuit Switching with Intelligent Flow Management," in Journal of Lightwave Technology, vol. 33, no. 5, pp. 1063 - 1071, 2015.

[14] M. C. Yuang *et al*., "OPMDC: Architecture Design and Implementation of a New Optical Pyramid Data Center Network," Journal of Lightwave Techn., vol. 33, no. 10, pp. 2019-2031, 2015.

[15] M. Imran, M. Collier and P. Landais, "HOSA: hybrid optical switch architecture for data center networks," *in Proc*. of *ACM ICCF*, Ischia, Italy, 2015.

[16] M. Imran, M. Collier, P. Landais and K. Katrinis, "Performance evaluation of hybrid optical switch architecture for data center networks," *Optical Switching and Networking,* no. 21, pp. 1 - 15, 2016.

[17] T. Tekin, N. Pleros and D. Apostolopoulos, "Photonic Interconnects for Data Centers," *in Proc*. of OFC Conference, San Francisco, 2014.

[18] K. Chen *et al*., "WaveCube: A Scalable, Fault-Tolerant, High-Performance Optical Data Center Architecture," *in Proc*. of IEEE INFOCOM, Kowloon, Hong Kong, 2015.

[19] G. A. Fish and D. K. Sparacin, "Enabling flexible datacenter interconnect networks with WDM silicon photonics," *in Proc*. of IEEE CICC, San Jose, California, 2014.

[20] H. Liu, F. L. Lam and C. Johnson, "Scaling Optical Interconnects in Datacenter Networks: Opportunities and Challenges for WDM," in Proc. of IEEE Symposium on HPI, Mountain View, California, 2010.

[21] Riverbed, "Steelcentral netplanner - network planning," Riverbed Technology, 2016. [Online].

[22] Alcatel-Lucent, "Alcatel-Lucent 1830 Photonic Service Switch (PSS-64 and PSS-36)," 2011. [Online].

[23] MRV Communications, "Optical Cross Connect: OCC 96," 2012. [Online].

[24] Finisar Corporation, "Pluggable Optics for the Data Center," 2016. [Online]. Available: https://www.finisar.com/.

[25] Arista Networks, "Arista 7320X Series Technical Specifications," 2016. [Online]. Available: http://www.arista.com/en/products/7300-series. [Accessed August 2016].

[26] Huawei Technologies Co., Ltd, "CloudEngine 5800 Series Data Center Switches," 2016. [Online]. Available: http://www.huawei.com.

Paper D: Combining hardware and simulation for datacenter scaling studies

S. Ruepp*, **A. Pilimon***, J. Thrane, Michael Galili, Michael Berger and Lars Dittmann, "Combining hardware and simulation for datacenter scaling studies", in *Proc. International Conference on Optical Network Design and Modeling (ONDM)*, Budapest, Hungary, 15-18 May, 2017, pp. 1-6.

ISBN: 978-3-901882-93-7

Combining Hardware and Simulation for Datacenter Scaling Studies

Sarah Ruepp*, Artur Pilimon*, Jakob Thrane, Michael Galili, Michael Berger, and Lars Dittmann

DTU Fotonik, Dept. of Photonics Engineering
Technical University of Denmark
{srru,artpil,jathr,mgal,msbe,ladit}@fotonik.dtu.dk

Abstract— **Datacenter networks are becoming crucial foundations for our information technology based society. However, commercial datacenter infrastructure is often unavailable to researchers for conducting experiments. In this work, we therefore elaborate on the possibility of combining commercial hardware and simulation to illustrate the scalability and performance of datacenter networks. We simulate a Datacenter network and interconnect it with real world traffic generation hardware. Analysis of the introduced packet conversion and virtual queueing delays shows that the conversion efficiency is at the order of a few microseconds, but the virtual queuing may have significant implication on the performance analysis results.**

***Keywords*— Datacenter, simulation, system-in-the-loop**

I. Introduction

Datacenters and datacenter networks (DCNs) are becoming increasingly important to our current and future communication infrastructure. Network connectivity is omnipresent from a multitude of devices, and they all generate data that must be stored and processed. This puts an enormous strain on datacenters and requires novel approaches to ensure scalability of the underlying DCN infrastructure.

Unfortunately, current DCN architectures are not easily scalable, and current solutions impose unsustainable overheads in terms of capacity, connectivity and energy consumption requirements [1]. It is thus essential to develop fundamentally new hardware technologies, internal datacenter network connection infrastructures joined with advanced mechanisms for control and service orchestration.

A challenge faced by researchers is that real world datacenters often are closed systems, and building up commercially-sized datacenters simply for research purposes is generally highly unfeasible from both an economic and a footprint wise perspective. On the other hand, simply conducting datacenter research by means of mathematical analysis or simulation does not visualise the effect of using real components (e.g. processing delays, setup constraints, timings, etc.), and may thus provide unrealistic results when concepts are to be ported to real networks. In other words, what may work on a single computer for research purposes may not be feasible in a commercial datacenter infrastructure.

In this work, we thus showcase the approach of combining hardware and simulation for datacenter performance scaling studies using Riverbed's System-in-the-Loop (SITL) tool [2]. This gives the possibility to use commercial components and their respective properties whenever available, and to evaluate different scalability and performance aspects without having to acquire enormous amounts of expensive components.

The remainder of this paper is organized as follows: section II deals with optical datacenter networks. In section III, the combined hardware and simulation setup is explained. Section IV presents the results and the paper is concluded in section V.

II. Optical Datacenter Networks

One of the challenges the datacenter industry is facing is to move away from vendor-specific, hierarchical, statically controlled and poorly scalable infrastructures. In the context of the EC FP7 project COSIGN [1], the partners pursue the development of scalable, low-latency, cost-effective, versatile multi-technology datacenter networks, that can combine the benefits of introducing optics and SDN (Software Defined Networking) into the datacenter ecosystem (shown in Figure 1).

Datacenter network architectures have traditionally been built over variations of the Fat-Tree, BCube, D-Cell, flattened butterfly, etc. [3]. Newer proposals contain a Ring-of-Rings (RoR) architecture [4], which benefits from a high internal robustness due to the inherent protective ring architecture.

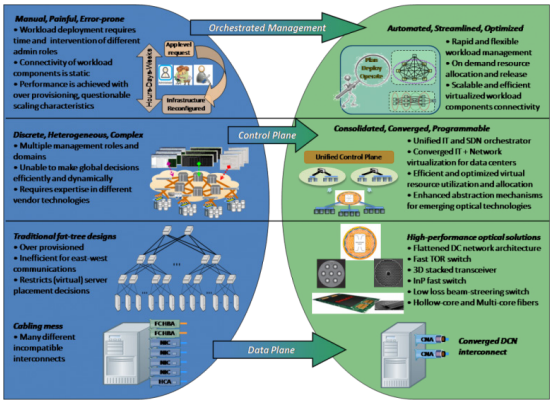


Figure 1: COSIGN datacenter evolution [1]

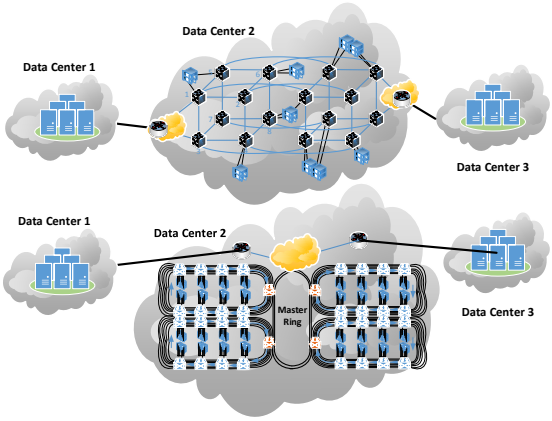


Figure 2: HyperCube and Ring-of-Rings architectures

Additionally, HyperCube and HyperX structures [5] have been widely used in the high performance computing (HPC) industry, and due to their benefits in terms of low latency and high scalability propose themselves as promising candidates for future datacenter architectures.

Furthermore, the introduction of optical components and optical switches can significantly increase the amount of data that can be transmitted within a datacenter. Figure 2 illustrates how Top of Rack (TOR) switches in datacenters are interconnected using either the Ring-of-Rings or the HyperCube topology in an inter-DC communication scenario.

III. System In The Loop Simulations

In order to facilitate the performance analysis and scaling studies of datacenters, simulation and especially integration of real hardware into the simulations, Riverbed's System-in-the-Loop (SITL) tool [2] provides some very powerful features.

The tool provides the linkage between the "real world" and the simulation that is running inside a computer. Any type of device, e.g., a router or a switch, can be linked to the simulation via the workstation's Ethernet port. Let's assume a packet that is generated by a real server, processed in a simulated datacenter network, and terminated by another real server. Once the packet is generated on the real server, it will be transmitted via Ethernet to the workstation running the simulation. Most importantly, the real time and the simulation time are running continuously, ensuring that timing is kept consistently throughout the entire system. The real packet is then converted into a simulation packet that will then be processed throughout the simulation. When the packet has passed through the desired simulation path, it is mapped at the external interface and converted back to a real packet. The real packet is then sent from the workstation through the Ethernet interface towards the real equipment. An abstracted packet translation procedure is illustrated in Figure 3.

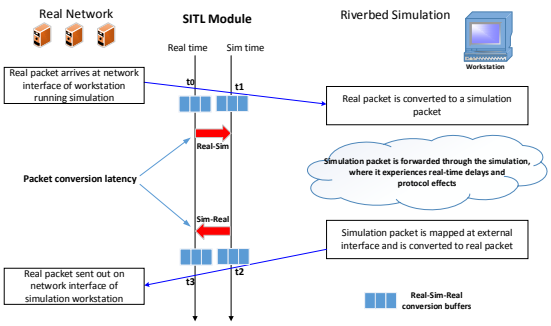


Figure 3: Packet flow between real and simulation equipment

IV. Simulation Scenario

In this work, the Ring-of-Rings and the HyperCube architectures are analysed. Unfortunately, having such a setup containing real datacenter hardware available for experimental and research purposes is often unrealistic due to high cost and space (footprint) requirements.

We are using two state of the art traffic generators, namely Xena Testers [7], to generate traffic on different layers between Datacenter 1 and Datacenter 3. The traffic is passed through Datacenter 2, which is modelled using an internal architecture of either HyperCube or Ring-of-Rings.

Every topology is composed of a multitude of TOR switches, each of them having multiple servers attached. Thus, the architectures are built as simulation models to illustrate scalability without having to acquire multiple TOR switches.

The packet translation efficiency is highly dependent on the translation level needed (see Figure 4). Some traffic may not be terminated within the DCN, hence certain payload types will be just copied as a block of bits, not touching the corresponding headers, resulting in faster translation. The simulation configuration (setup) and real hardware is shown in Figure 5, where the simulation model and real equipment are linked via the aforementioned virtual SITL gateways.

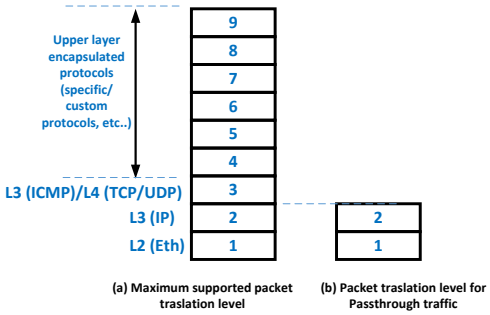


Figure 4: Packet translation depth (level) at the real/simulated interface

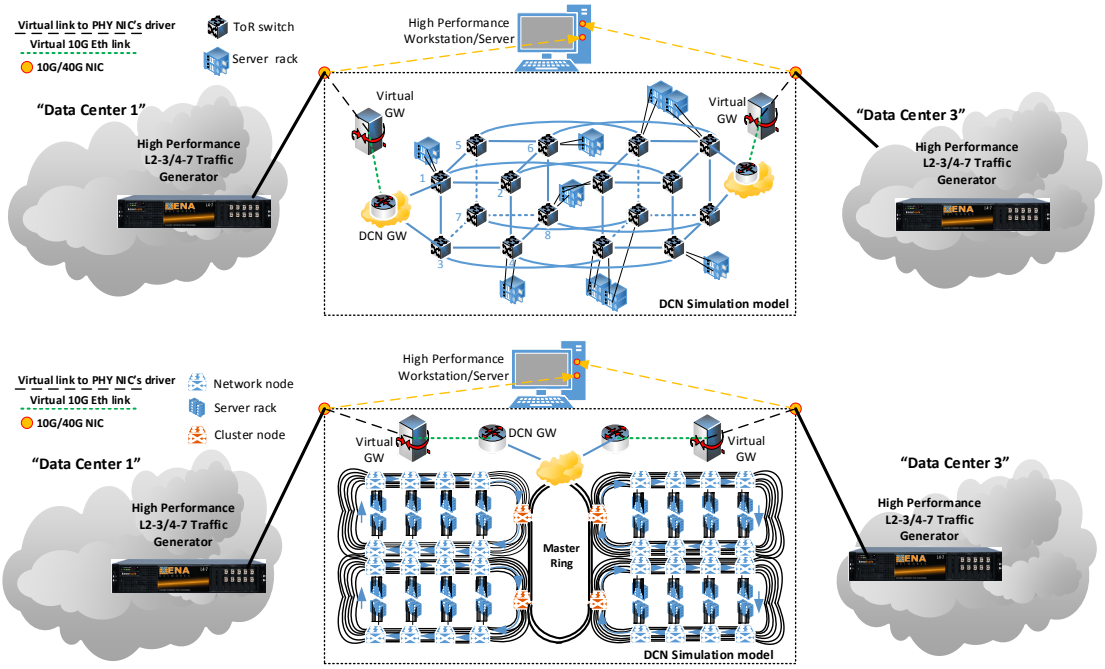


Figure 5: Experimental setup combining real equipment and simulation

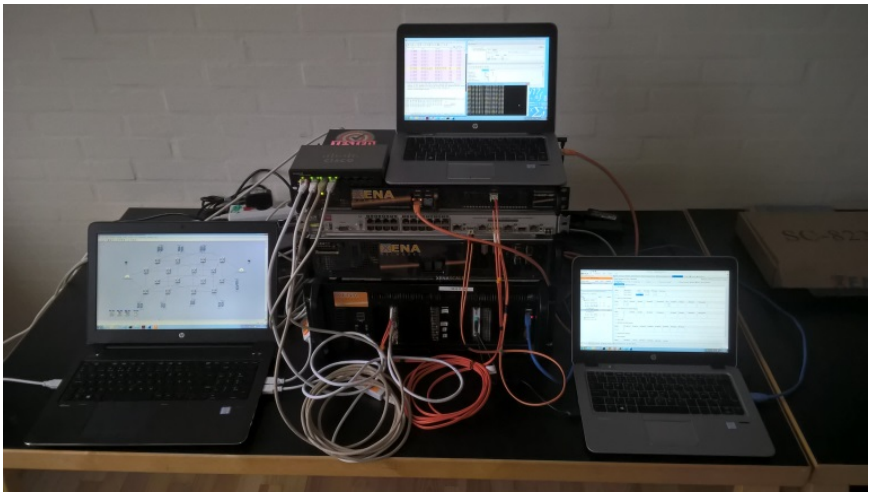


Figure 6: Picture of the experimental setup

The entire system setup is shown in Figure 6. Note that in the depicted experimental setup we used portable computers for the initial demonstration tests. In this case a maximum rate of 80 and 120 Mbit/s was achieved in our tests for the ICMP and TCP traffic, respectively, on a 1GE network interface due to encountered Windows socket buffer overflow (operations on non-blocking sockets that cannot be completed) as described in [8]. Large scale experiments will run on a dedicated set of more powerful servers. The "workstation" on the left (black) is running the simulation model of datacenter 2. The Xena testers in the middle are emulating datacenter 1 and datacenter 3 by generating different predefined traffic patterns.

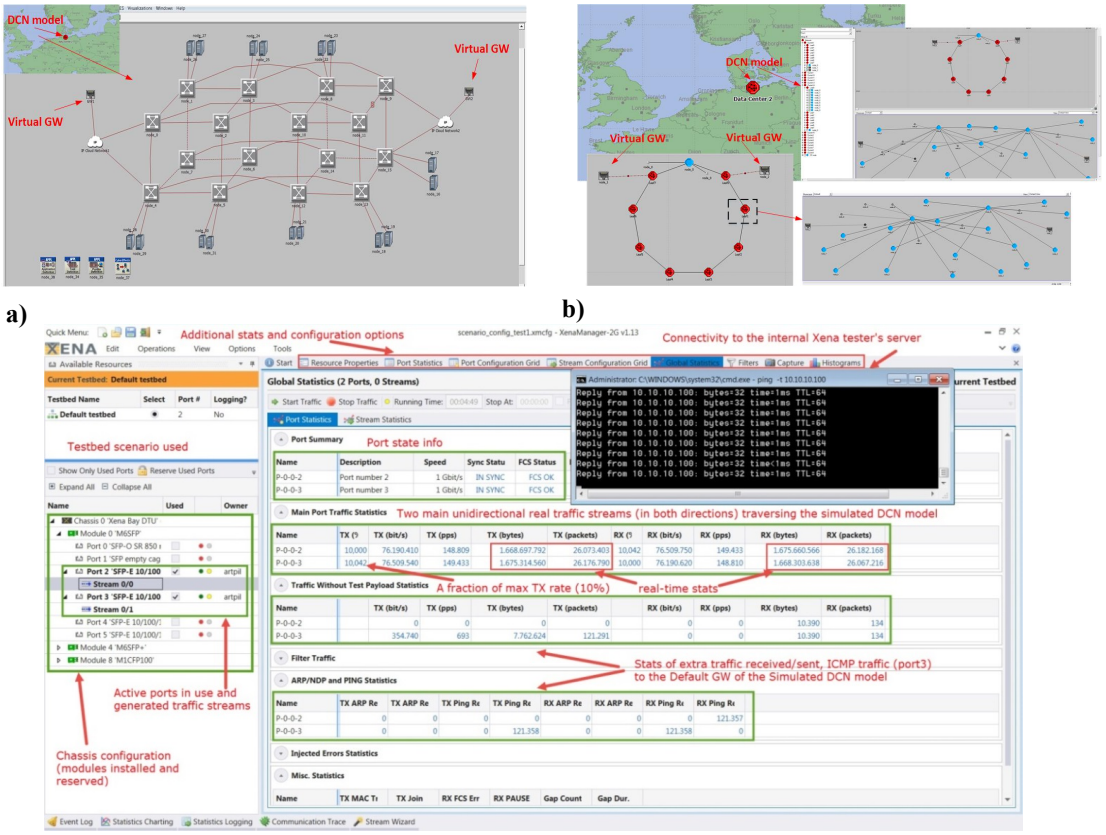


Figure 7: Software components of the testbed setup: a) Simulation model of HyperCube-16 architecture; b) Simulation model of Ring-of-Rings architecture; c) Picture of Xena L2-3 tester

The main benefits of using these high performance testers are: a) possibility to generate realistic application layer communication patterns (*pcap*-based replay); b) possibility to emulate millions (when we need scale) of concurrent traffic flows by utilizing multiple available transceiver modules (1GbE, 10GbE, 40GbE and 100GbE rates, depending on the modules used). For a realistic experimental case study it is sufficient enough to have a few 10G or 40G modules, since present day workstations or servers can be equipped with 10G or even 40G NICs (Network Interface Card) to provide a reasonable uplink/downlink for our modelled DCN.

The laptop on top of the system is analysing packets running a Wireshark tool. The software for the Xena testers is executed on the laptop on the right side of the picture.

Detailed screenshots from the individual computers and models can be seen in Figure 7, showing the simulation models and Xena tester software interface, respectively.

It is important to point out that evaluation of the packet translation latency parameter at the real-simulated network interface is crucial in the context of DCN performance analysis, because of much more stringent timing requirements, compared to conventional networks. In modern (and future) DCN environments there are several critical factors, which set DCNs aside into a different "networking" category, namely ultra-low latency and high throughput requirements, ultra-short duration of vast majority of the intra-DC traffic flows (in the order of a few 10s or 100s of *ms*), significantly larger east-west (internal, remaining within a DC) traffic volumes as compared to south-north (external) traffic. That is the reason for assessing the feasibility of using such a hybrid system first.

V. Results

One of the most important measures in datacenter networks is latency. We thus measure the time that it takes to traverse the SITL gateway nodes in both directions, namely the conversion delay on this virtual interface. This parameter is of paramount importance when it comes to the further performance evaluation of the simulated DCN topology in terms of delay, because it directly affects the accuracy of the obtained measurement results.

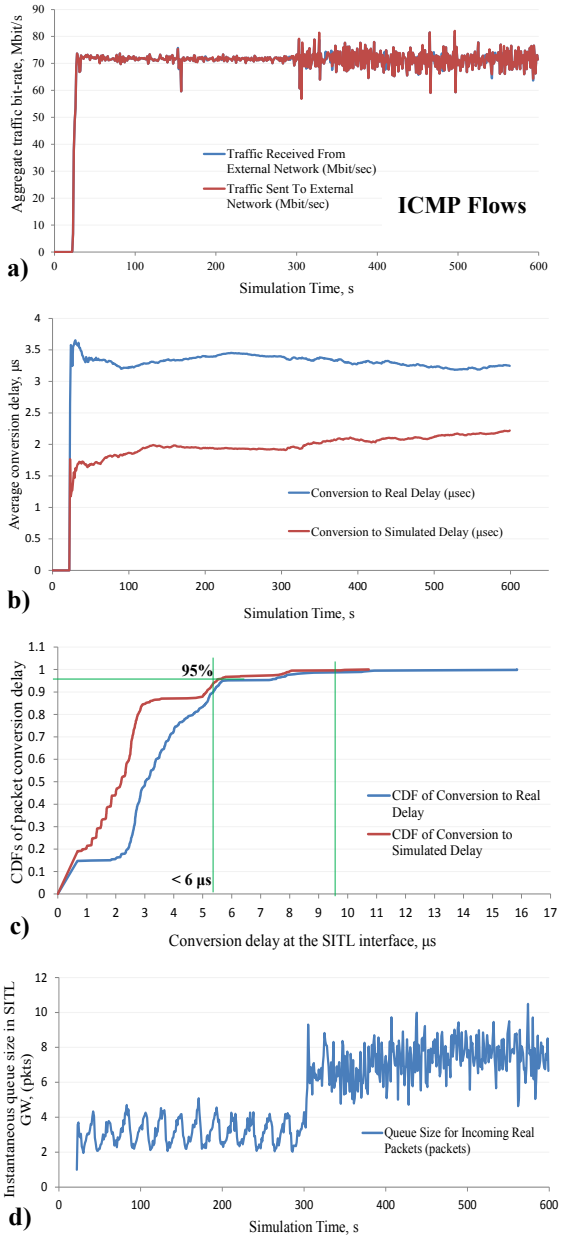


Figure 8: Results of Simulation model's stress-testing. Packet conversion efficiency at the SITL GW interface. Test1

The stress-testing was performed by loading (symmetric bidirectional traffic) the transit (simulated) DCN, datacenter 2, with a large number of high bit-rate ping flows, launched sequentially using a script. The results are illustrated in Figure 8. We observe that under the load of ~ 75-80 *Mbit/s* (see Figure 8*a*), the time-average conversion delay for the incoming traffic (real-to-simulated) is fluctuating around 2.0 *µs* per packet, while in the opposite direction (simulated-to-real) it is almost 60% higher. However, this is a relative difference, since this result shows a time-average value. It's more interesting to look at the Cumulative Distribution Function (CDF) of this parameter, being a better indicator. As it can be seen in Figure 8 (*c*), for around 95% of collected samples, the per packet conversion delay is less than 6 *µs* on average for both directions, with the worst case scenario being below 10 µs. As a result this penalty must be taken into consideration while evaluating the performance metrics of a DCN under consideration. The evolution of the queue size in virtual SITL gateway is shown (Figure 8*d*) to be relatively low (3-8 packets on average), but this parameter is very important, since it will affect the queueing delays under much higher traffic loads.

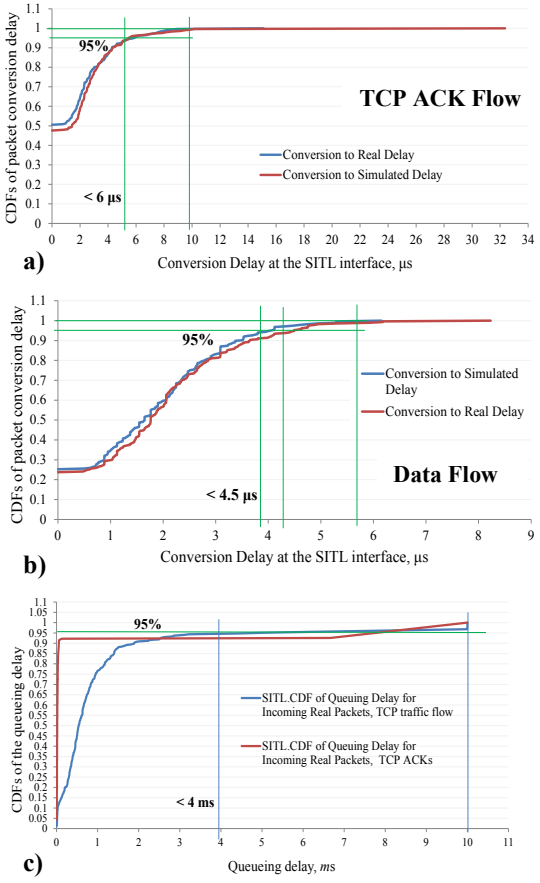


Figure 9: Results of Simulation model's stress-testing. Packet conversion efficiency at the SITL GW. Test2

In another test the SITL gateways were loaded with asymmetric traffic flow, namely by emulating a transmission of a large data file (3.41 *GB*) via TCP connections (data in the forward direction, streams of ACKs in the reverse) with the average data rate of 120 *Mbit/s*. Transmission profile followed a bursty traffic profile configured.

We analysed the CDFs of bidirectional packet conversion and queueing delays, presented in **Figure 9** (*a* – CDF for the TCP ACK flow, *b* – Data traffic flow, *c* – queueing delays for the incoming real packets). As can be seen, statistically, the conversion delay is at the order of a few *µs* in both directions, and 95-percentile latency is in the same range for both flows. However, considering the proportion of packets corresponding to each flow (Data and TCP ACKs), the average number of Data packets per second (pps) was around 14000, whereas for the TCP ACK stream it was ~ 50% of that, namely around 7000 pps. This proportionality is expected, since by default TCP connections were using a *delayed ACK* mechanism. Thus, this dependency is reflected in the conversion delay results in **Figure 9** (*a*) and (*b*), where the conversion delay of the 50% of the TCP ACK packets is less than 1 µs, while 50-percentile of the data packets experience delays twice as large (~ 2 µs).

When it comes to the packet queueing in SITL, **Figure 9** (*c*) shows the delay experienced by more than 92% of ACK stream packets is under 1 *ms* (at the order of a few *µs*), while data packets are queued up to 4 *ms* (95-percentile), with the worst case scenario of ~ 10 *ms*. The latter values are relatively high and will have a serious impact on the performance results.

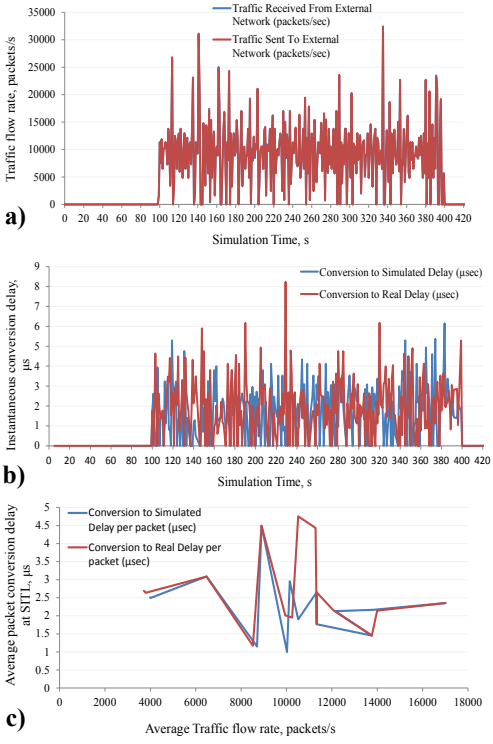


Figure 10: Dependency of the packet conversion latency on the traffic flow rate in virtual SITL gateway node

We evaluated the dependency of the packet flow rate on the average conversion delay by statistically sampling the packet rates and corresponding (by simulation timestamp) conversion latency using the obtained distributions (Figure 10 *a, b*) and the preliminary results (see Figure 10 *c*) show that there is no clear link between the packet rate and conversion delay incurred, and the stochastic nature may be a result of several other factors, such as specifics of packet capture by the WinPcap [9] (libPcap for Linux) module, implementation of the conversion functions (code) and characteristics of the NIC installed (buffering, protocol checksum offload, etc.).

VI. Conclusion

In this paper, we have detailed an approach for combining real hardware and simulation for the purpose of evaluating the performance and scalability of datacenter networks. We describe how the Riverbed System-in-the-Loop (SITL) tool can be used to interconnect real world and simulation under continuous timing constraints, without having to invest in vast amounts of expensive hardware. Our results show that the SITL gateway adds a conversion delay in the order of microseconds as well as load-dependent buffering delays that must be taken into consideration for any latency measurements.

The approach presented here, showcasing the combination of real hardware and simulation, has an enormous potential in the emerging integration of optics in the datacenter world, where scaling and latency effects must be studied without necessarily having access to real datacenter infrastructures.

Acknowledgment

This work has been partially supported by the EC FP7 project “COSIGN, grant no. 619572”, and the Innovation Fund Denmark project “Layer 4-7 Testing at 100 Gbps”.

References

[1] EU FP7 Project Cosign, Combining Optics and SDN In next Generation data centre Networks, http://www.fp7-cosign.eu/

[2] Riverbed System in the Loop Tool (SITL). www.riverbed.com (formerly known as OPNET Inc.)

[3] D. Abts, J. Kim. “High Performance Datacenter Networks: Architectures, Algorithms, and Opportunities”, in Synthesis Lectures on Computer Architecture (2011)

[4] A. M. Fagertun, M. Berger, S. Ruepp, V. Kamchevska, M. Galili, L. K. Oxenløwe and L. Dittmann (2015). “Ring-based All-Optical Datacenter Networks”, in Proc. of European Conference on Optical Communications (ECOC) , Valencia, Spain

[5] J. Ho Ahn, N. Binkert, A. Davis, M. McLaren, R. S. Schreiber. “HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks”, in Proc. of the Conference on High Performance Computing Networking, Storage and Analysis - SC '09 (2009)

[6] SITL session, Opnetwork conference 2010, www.opnetwork.com

[7] Xena Networks, www.xenanetworks.com, Xena Bay L2-3 and Xena Scale L4-7 Testers

[8] Microsoft, “Windows Sockets Error Codes,” 2017. [Online]. Available: https://msdn.microsoft.com/en us/library/windows/desktop/ms740668

[9] The Windows packet capture library, https://www.winpcap.org/.

Paper E: A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks

A. Pilimon and S. Ruepp, "A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks", presented at *IEEE International Conference on Computing, Networking and Communications (ICNC)*, Maui, Hawaii, USA, March 5-8, 2018, pp. 1-5.

DOI: -

A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks

Artur Pilimon *and* Sarah Ruepp
Department of Photonics Engineering
Technical University of Denmark
{artpil, srru}@fotonik.dtu.dk

***Abstract*— Datacenters (DC) as well as their network interconnects are growing in scale and complexity. They are constantly being challenged in terms of energy and resource utilization efficiency, scalability, availability, reliability and performance requirements. Therefore, these resource-intensive environments must be properly tested and analyzed in order to make timely upgrades and transformations. However, a limited number of academic institutions and Research and Development companies have access to production scale DC Network (DCN) testing facilities, and resource-limited studies can produce misleading or inaccurate results. To address this problem, we introduce an alternative solution, which forms a solid base for a more realistic and comprehensive performance evaluation of different aspects of DCNs. It is based on the System-in-the-loop (SITL) concept, where real commercial DCN equipment (switches) is interconnected with simulated DCN nodes via the SITL virtual gateway modules in the Riverbed Modeler, forming a unified hybrid real-simulated DCN setup. This testbed is complemented with high performance network testers, which can generate a mix of different real traffic streams at L2-3 and L4-7 at the rates of 1, 10 or up to 40 Gbps. These components allow performance benchmarking tests to be conducted at large scale.**

Keywords—Hybrid Datacenter Testbed; System-in-the-Loop; real-time simulation; large-scale Datacenter

I. INTRODUCTION

IT and Communication Technologies (ICT) have become the backbone of the global economy, and Datacenters (DC) as well as associated intra- and inter-DC networks are playing a role of mission critical infrastructures. Ubiquitous network connectivity, increasing global user base, new feature-rich interactive services and applications – these are some of the key reasons, affecting the global growth of the data traffic volumes, most of which constitute internal DC traffic (East-West traffic), originating and terminating within a DC [1]. As a result, existing as well as next-generation DC infrastructures are being seriously challenged in terms of the resource utilization and energy efficiency, scalability, availability, reliability and performance requirements (ultra-high bandwidth and ultra-low latency).

In order to address the aforementioned challenges, various optimizations and innovative solutions are being proposed by the research community. However, in many of the cases, these customized or new algorithms, protocols and network architectures are evaluated in a very small-scale testing environment with very specific use-cases or in a functionally limited simulation or emulation tool. The outcome of that is such that the obtained results may be inaccurate or misleading, and many innovative solutions and technologies, incubated in the lab environment, may never be transferred to large-scale industrial deployments. Therefore, it is important to point out some common main reasons why some research ideas are not getting proper attention and presented results are often incomplete. First, lack of convincing measurements and analysis, which are applicable at scale, is the main barrier for the transfer of innovative research. Second, assumptions and simplifications, introduced in the design phase of an experiment or a simulation model due to lack of resources, such as uniformly distributed traffic, no realistic background traffic or network processing effects, no possibility to deploy a real DC- or cloud-native application or service to test, lead to discrepancies in the results, compromising their credibility. Third, there are often two extremes of resource-limited research, namely only experimental validation using a small-scale test setup of hardware, which can hardly be extended to emulate a large-scale context, or a simulation-based analysis, which can scale better and can be reproduced, but suffers from unavailability of DC-specific traffic sources (real servers with fully-featured applications) and realistic usage patterns.

A way to rectify such a situation would be to perform the initial evaluation and small-scale testing of the developed ideas as a Proof-of-Concept (PoC) in a local research lab, and then to access a production scale DCN testing environment in order to conduct an extensive large-scale validation. However, only a limited number of research groups have access to such high-end testing facilities, and building up anything similar in a local research lab is unrealistic both financially and timewise.

In this work we introduce an alternative solution, which could form a solid base for a more realistic and comprehensive performance evaluation of different aspects of DCNs. It is based on the System-in-the-loop (SITL), also known as Hardware-in-the-loop (HIL), concept, where real commercial DCN equipment (e.g., switches) is interconnected with simulated DCN nodes via the SITL virtual gateway modules in the Riverbed Modeler [2], forming a unified real-simulated DCN setup. This testbed is complemented with high performance network testers, generating a mix of different traffic streams at L2-3 and L4-7 at the rates of 1, 10 or up to 40 Gbps. The PoC of building and feasibility of using such a hybrid setup for DCN performance studies was demonstrated in [3], where the communication of high performance L2-7 network testers and software traffic generators through a

simulated DCN environment was assessed. The main contributions of this work are the following: 1) a hybrid (real-simulated) DCN testbed, consisting of real commercial and simulated network switches, and high performance hardware traffic generators (network testers) for more realistic stress-testing; 2) an outline of potential application scenarios of using such a testbed for large-scale experimental scalability and performance studies.

The remainder of this paper is organized as follows: section II presents the state of the art hybrid solutions for the DCN-specific large-scale studies. In section III, the hybrid DCN hardware and simulation setup is explained. Section IV outlines the use-cases considered for large-scale DCN studies using this setup and the paper is concluded in section V.

II. Related Work

There are different types of network research testbeds proposed, which can be categorized into local physical [4], emulation-based [5][6] and hybrid [7][8] testbeds, as well as globally distributed purely virtualized testbeds such as PlanetLab [9] or OneLab [10]. These experimental network research environments have been built to enable more comprehensive studies of a larger spectrum of network communication problems. However, the focus of this overview is on the existing combined (using a combination of hardware and some form of emulation or simulation techniques) systems, created for large-scale experimental studies of DCN environments, where the scalability and functional capabilities of the setup are the most important features. Therefore, the aforementioned testbed setups [4-10] are out of the scope of this discussion, because they are not specifically targeting DCN-oriented large-scale environments, and a DCN-specific setup, presented in [11], will be described next.

Benet in [11] recently proposed a Cloud testbed, called OpenStackEmu, which is built as a combination of a well-known OpenStack platform, a CORE (Common Open Research Emulator) network emulator and an external SDN (Software Defined Networking) controller [11]. The real-time network emulator uses virtual TUN/TAP (network TUNnel and tap) interfaces (virtual network kernel devices) in order to connect external physical devices to it and to be able to inject external traffic into the emulated DCN. The presented test setup consists of a set of external physical devices, connected to the machine running the CORE emulator: a basic set of OpenStack infrastructure nodes, as well as an external node with an SDN controller and one more node with background traffic generation software. This hybrid system allows testing various realistic DC networking scenarios with VM deployment, live VM migration, service orchestration and it can be used to study the impact of the DCN topologies, network characteristics and other aspects. However, it is not obvious how well such an emulated setup would scale, because the network devices (Open virtual Switch, OVS) as well as hosts are emulated using the OS-level virtualization principle, where each virtual device is modelled as a light-weight Linux container (represents a VM). This approach is more resource-demanding than just a simulated counterpart, and that impacts the scalability of the setup. Our work is different in such a way that the DCN itself is composed of real physical (electronic and one optical) and simulated network switches, which are communicating via a set of electrical-optical interfaces, installed on the high performance server, hosting the simulation environment with modelled devices.

III. A Hybrid DCN Testbed Architecture

A high-level architecture of the hybrid real-simulated testbed setup is shown in Figure 1. As it can be seen, the setup consists of a complete (symmetric) 16-Hypercube structure, half of which is composed of 8 commercial HP Aruba ToR (Top of Rack) switches, and half is simulated in a Riverbed Modeler tool with real-time simulation kernel and using virtual SITL gateway modules for connectivity to the external physical hardware (or software processes). The important components, differentiating this hybrid setup from the others, are summarized in Table I.

TABLE I. Key functional elements of a DCN Testbed

Functional Element(s)	Purpose
Simulation platform with SITL	Virtual hardware-software linking for real-time communication with real DC equipment
High performance Xena Bay/Xena Scale network testers [12]	Accurate generation of large volumes of data with mixed traffic patterns, many simultaneous connection-oriented and connection-less flows. Creating a more realistic DCN communication context, difficult to achieve with software gen.
Integration of simulated, electrical and optical DC network equipment	Possibility to evaluate the scalability of different DCN architectures by stress-testing with real communication flows; assessment of the benefits of optical switching in hybrid DCN architectures.

A short summary of the technical characteristics of this hybrid DCN setup is provided in Table II. However, these technical specifications are not definitive and any combination of powerful enough hardware can be used. The same applies to the choice of a DCN topology: we modelled and assembled this topology, because Hypercube, HyperX, Flattened Butterfly and other direct-connection structures [13] have been widely used in HPC (High Performance Computing) environments due to their well-known features such as high scalability and low latency. Such flattened, high-radix topologies in combination with optical switching may be considered as promising candidates for the next-generation large-scale architectures.

The “physical” part of the DCN structure, namely one 8-node sub-cube (Figure 1, *left*), has two types of physical layer connections: interconnections with the physical and simulated ToR switches within the real-simulated 16-Hypercube using Multi-Mode Fiber links (MMF) and optical shortcut connections through Polatis 24x24 free-space Optical Circuit Switch (OCS) using Single-Mode Fiber (SMF) links.

The simulation environment is hosted on a high performance server (Figure 1, *right*), capable to run a large-scale simulation model with multiple virtual SITL gateway modules to connect to the external devices via available electrical-optical interfaces. In this particular test setup, we used IBM 2U X3690 X5 server with Intel Xeon X5650 2-CPU (24 logical cores), 128 GB RAM and 4 installed PCI-e NICs (Network Interface Card) with up to 4 ports on each.

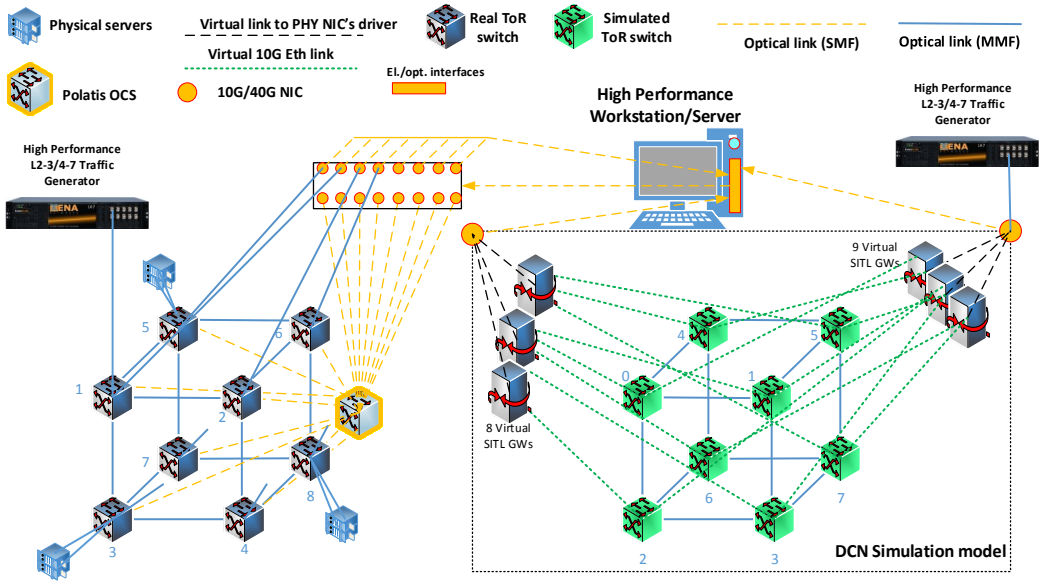


Figure 1 A high level architecture of a hybrid DCN research testbed

Please note that on Figure 1 all electrical-optical interfaces of the server machine are shown on 1 board (with 16 ports) for presentation clarity. The ToR-simulation-server connections of the bottom plane of the real 8-Hypercube (3, 4, 7 and 8 switches) are also not shown for the visual clarity of the presentation, but these connections are present.

TABLE II. PERFORMANCE/FUNCTIONAL CHARACTERISTICS OF A DCN TESTBED

Network node type	**Characteristics**			
	Network interface type/rate (Gbps)	***Nr. El./opt. interfaces***	***Nr. Optical interfaces***	***Optical switching capabilities***
ToR switches	SFP/SFP+, 1/10G	16	-	-
Polatis OCS	Signal bit-rate/format independent up to 100G, 400G and beyond	-	48	OCS, Fiber Cross connect
Network testers	SFP+/1-10G, QSFP/CFP – 40/100G	6-12+ (custom.)	-	-
High Perf. Server	SFP/SFP+, 1-10G	Up to 16[a]	-	-

[a] For IBM X3690 X5 Intel Xeon X5650 24-core machine, with 4 PCI-e slots, 128 GB RAM

The network switches of the simulated 8-node sub-cube (Figure 1, *right*) maintain two types of logical connections: regular simulation logical links between simulated ToR switches and *virtual SITL 10Gbps links*. The latter are used by the simulation kernel to create a logical channel from the simulated Ethernet-enabled device (e.g., a router, a switch or a workstation) to the driver of the physical NIC's port to be able to perform bidirectional real-time extraction and conversion of the data packets (Ethernet frames). The performance of the simulated environment can be further enhanced by splitting a large-scale simulation model into clusters and parallelizing the setup for even larger scaling. In addition, the use of Multi-Threading (MT) of the simulation model within each cluster can further increase performance and reduce such critical parameter as packet conversion and buffering latency, which was thoroughly investigated in the previous work in [3].

The simulation model of an 8-node sub-cube of the hybrid DCN is depicted in Figure 2, where we can observe multiple virtual SITL modules, associated with the corresponding simulated nodes. The scalability aspect of this hybrid setup is reflected in the capacity of the simulation environment to scale to large DCN topologies, which is only subject to the hardware limitations of the server nodes used. As it can be seen in Figure 3, we can create a sophisticated DCN architecture of a larger scale in the simulation environment and complement it with a smaller set of real DCN equipment. In this scenario we created a model of asymmetric (incomplete) Hypercube by combining a 44-node Hypercube with 4 additional physical commercial DCN switches, forming an incomplete (deemed to be more efficient from the scalability point of view) 48-node Hypercube. Moreover, it has 4 fast optical shortcut connections through Polatis OCS.

Integration of the Optical Circuit Switch (Polatis) with the hybrid testbed was performed to be able to study the potential benefits (relative gain in terms of latency reduction and increase of throughput) of introducing optical switching in DCNs that can be assessed more accurately in larger test network scenarios.

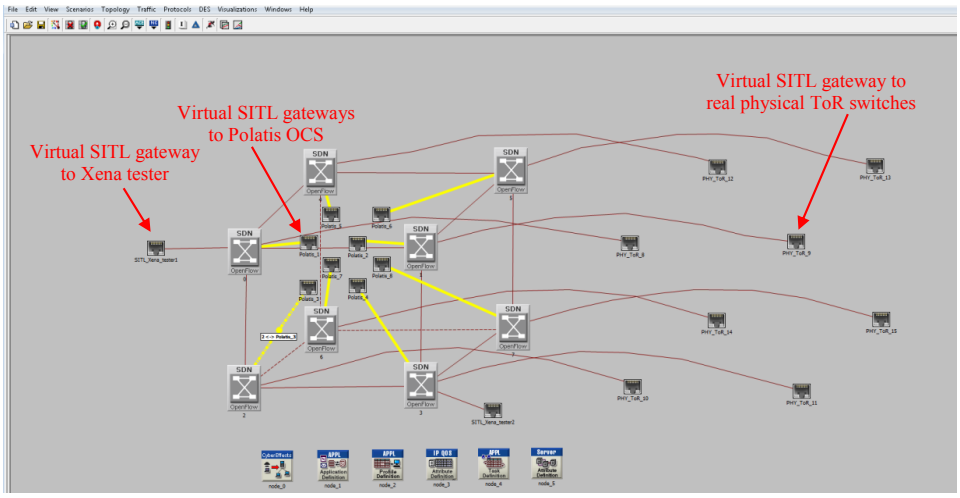


Figure 2 A SITL simulation model of 8-Hypercube structure with connections to the physical 8-Hypercube, a Polatis OCS and Xena testers

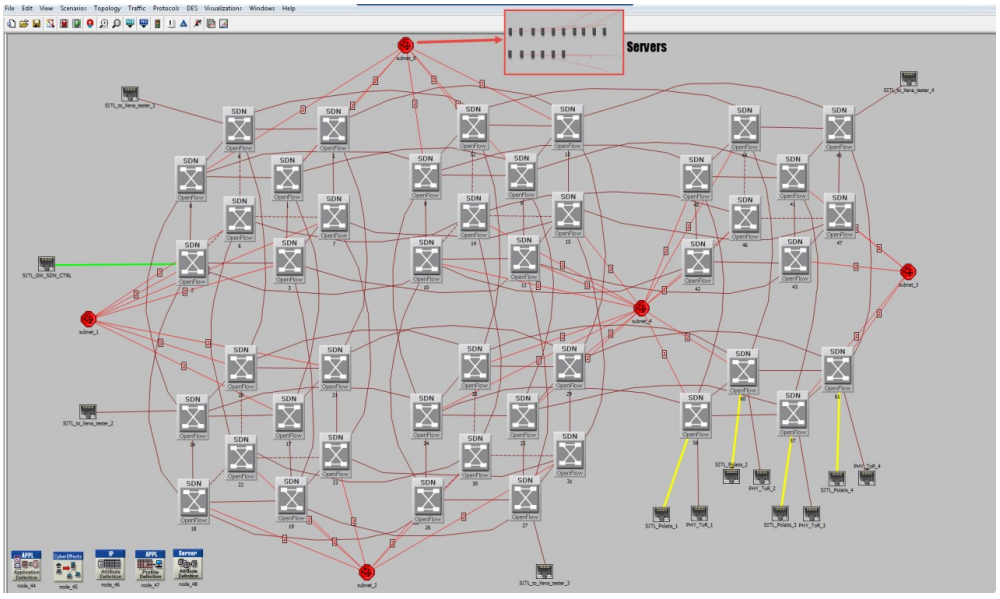


Figure 3 A SITL simulation model: 44-Hypercube, connectivity to Polatis OCS, connectivity to physical ToR switches and Xena network testers

Preliminary testing was conducted by measuring the end-to-end communication latency (using the capabilities of Xena testers) in the hybrid real-simulated Hypercube in two settings: with Polatis OCS fast optical shortcuts and without Polatis OCS. The results are shown in Figure 4, and, by using fast reconfigurable optical shortcuts, the latency, even in a 16-switch DCN setup, can be reduced by more than 50%, considering a long 6-hop path spanning the Simulated switches (0, 1, 3, 7) and Real switches (1, 2, 6) as compared to a short path switch_0 (simulated) ↔ Polatis OCS ↔ switch_1 (real). Please note that this measurement result shows a normalized and 1-second averaged latency to show the relative gain of an optical bypass circuit when steering the traffic.

Additional clusters of simulated DC servers are added to the simulation model to create background traffic within the modelled part of the DCN, based on user-defined demands. Finally, the presented testbed is also SDN-ready (Software Defined Networking), since the commercial HP Aruba ToR

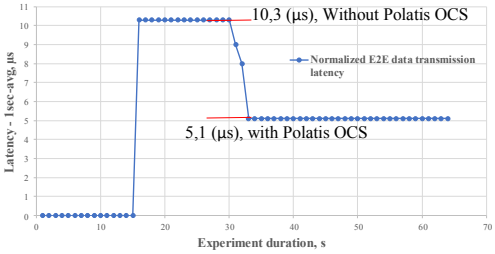


Figure 4 Normalized latency measurement in a hybrid DCN testbed

switches, simulation switches as well as Polatis OCS are all OF-enabled (Open Flow protocol).

IV. Testing Scenarios and Scalability Aspects

Examples in Figures 2 and 3 show that we can examine a large number of communication scenarios under more realistic networking constraints. The following (but not limited to) use-cases and DCN-related aspects can be studied using this setup:

- ***Real servers and simulated DCN***. A small cluster of real application servers can be deployed, running a set of test applications (e.g., multi-tier services), which can be used for more accurate performance and quality of service characterization. For instance, a video streaming server can be deployed to assess the impact of different DCN parameters on the Quality of user experience. That would be more difficult to achieve in a small hardware-only setup (no DC scale) or in a simulation-only setup, where we cannot observe the video quality.
- ***Real servers, simulated DCN and Xena testers***. The same tests can be performed in a more realistic setting by specifically overloading certain parts of the DCN using high performance testers. That would be difficult to achieve using generic software traffic generators, e.g., if we would like to stress-test a DCN, which is equipped with 10/40 Gbps links. In addition, Xena testers support traffic generation using capture-replay method, which allows recording real communication sessions (e.g., skype, web browsing) and replaying them to many concurrent flows.
- ***Optical Circuit Switch and simulated DCN***. There is a large potential for the emerging integration of optics in the Datacenter world, where scaling and latency effects must be studied carefully. The latency reduction gain can be assessed more accurately in the context of a large-scale DCN environment while using fast optical shortcut links to offload the bottleneck areas of the network (e.g., separating elephant from mice flows).
- ***Simulated-Real-Simulated***. Inter-DCN communication aspects can be studied by combining two simulated DCNs with a real edge device (or a set of edge routers). For instance, performance aspects of geographically distributed DCNs can be analyzed using this method.
- ***Integration with SDN***. This scenario offers great opportunities for comprehensive analysis of different traffic routing/forwarding/engineering mechanisms.

V. Conclusion

In this work, we present an approach for building a hybrid DCN testbed for large-scale performance and scalability analysis. That gives a possibility to exploit realistic communication patterns under mixed traffic flows, to stress-test particular areas of a DCN topology of interest by combining real DCN hardware with simulated equipment and high performance network testers. The setup is used to evaluate the introduction of a fast Optical Circuit Switch in terms of latency, allowing studying the benefits of optical switching in a hybrid DCN context, as well as supports SDN-based control. Further non-exhaustive use-cases are presented as well.

Acknowledgment

This work has been partially supported by the EC FP7 project “COSIGN, grant no. 619572”.

References

[1] Cisco Systems, Inc., “Cisco Global Cloud Index: Forecast and Methodology, 2014–2019,” Cisco, 2015.

[2] Riverbed Technology, “Riverbed Modeler,” [Online]. Available: http://www.riverbed.com/products/performance-management-control/network-performance-management/network-simulation.html

[3] S. Ruepp, A. Pilimon, J. Thrane, M. Galili, M. Berger, and L. Dittmann, “Combining Hardware and Simulation for Datacenter Scaling Studies,” in *Proc. 2017 IEEE 21st International Conference on Optical Network Design and Modeling, ONDM*, 2017.

[4] S. K. Dhurandher, I. Woungang, I. Uppal, H. Bhanushali, and D. Gupta, “A Dot Net Framework based Physical Testbed for Ad hoc Network Routing Protocols,” *in Proc. 2011 24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2011, pp. 229–233.

[5] R. Lübke, D. Schuster, and A. Schill, “Reproducing network conditions for tests of large-scale distributed systems,” *in Proc. - 13th IEEE/ACM Int. Symposium on Cluster, Cloud, and Grid Comput.*, 2013, pp. 74–77.

[6] Q. Le-Trung, “Towards an IoT network testbed emulated over OpenStack cloud infrastructure,” *in Proc. - 2017 International Conference on Recent Advances in Signal Processing, Telecommunications and Computing*, 2017, pp. 246–251.

[7] A. S. Leger, J. Spruce, T. Banwell, and M. Collins, “Smart grid testbed for Wide-Area Monitoring and Control systems,” *in Proc. - IEEE Power Eng. Society Transmission and Distribution Conf.*, July 2016, pp. 1–5.

[8] H. Gao, Y. Peng, K. Jia, Z. Wen, and H. Li, “Cyber-Physical Systems Testbed Based on Cloud Computing and Software Defined Network,” *in Proc. - 2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IIH-MSP 2015, 2016, pp. 337–340.

[9] PlanetLab Consortium, “PLANETLAB: An open platform for developing, deploying and accessing planetary-scale services,” [Online].

[10] OneLab Consortium, “OnaLab: Future Internet Testbed,” [Online]. Available: https://onelab.eu/.

[11] C. H. Benet, R. Nasim, K. A. Noghani, and A. Kassler, “OpenStackEmu - A Cloud Testbed Combining Network Emulation with OpenStack and SDN,” *14th IEEE Consum. Commun. Netw. Conf.*, pp. 566–568, 2017.

[12] Xena Networks, “L2-3 (Xena Bay) and L4-7 (Xena Scale) Test Platforms,” [Online]. Available: http://www.xenanetworks.com.

[13] D. Abts and J. Kim, “High Performance Datacenter Networks: Architectures, Algorithms, and Opportunities,” *Synth. Lect. Comput. Archit.*, vol. 6, no. 1, pp. 1–115, Mar. 2011.

Paper F: A Novel Hybrid Testbed Combining Optics and SDN for Topology-Agnostic Datacenter Network Evaluation

A. Pilimon, S. Ruepp and L. Dittmann, "A Novel Hybrid Testbed Combining Optics and SDN for Topology-Agnostic Datacenter Network Evaluation", submitted to *Computer Networks Journal*, pp. 1-5, April 2018.

DOI: -

A Novel Hybrid Testbed Combining Optics and SDN for Topology-Agnostic Datacenter Network Evaluation

Artur Pilimon, Sarah Ruepp and Lars Dittmann

[1]

***Abstract*—Large-scale datacenter performance evaluation studies are often hindered by the absence of sufficient and functional hardware resources. This important aspect is limiting the diversity and completeness of the research results, as well as prohibiting faster transition of promising research ideas from the development setup in the lab to a large-scale deployment and wider acceptance in the DCN community. This work, therefore, proposes a hybrid physical-simulated electrical-optical and SDN-controlled testbed to address the challenges of large-scale performance and scalability characterization of Datacenter Networks with optical switching. Performance evaluation results show reasonable packet queueing and intra-simulation processing latencies, under 25 µs and under 160 µs, respectively. Moreover, a Software-Defined control and programmability of the setup allows for flexible scaling of Datacenter network model in a topology-independent manner, in addition to powerful performance measurement capabilities and testing accuracy.**

***Index Terms*—Datacenter Network, Hybrid Testbed, simulation, SDN, scalability**

I. Introduction

FAST proliferation of new service and communication models (e.g., Big Data Analytics, cloud computing and networking) facilitated tremendous increase in data traffic volumes. That inevitably leads to the emergence of more complex traffic patterns with diverse characteristics, resulting in application- and scale-specific performance requirements. This translates into much higher requirements for scalability, resource provisioning flexibility, energy efficiency, availability and reliability of Datacenter Network (DCN) infrastructures.

To tackle these challenges, different DCN architectures have been proposed, ranging from hybrid [1] (with electrical and optical switching) to all-optical architectures [2]. In addition, application of Software Defined Networking (SDN) in DCNs with optical switching offers great capabilities for automation, programmability, intelligent network resource orchestration, traffic engineering and advanced network virtualization [3].

One of the main challenges faced by new, innovative solutions for DC networking, is lack of diverse and comprehensive testing and performance evaluation at larger scale. One of the main reasons of that is generally limited availability of sufficient DC-scale testing resources (network infrastructure and realistic traffic models). This important aspect is prohibiting faster transition of promising research ideas from the development setup in the lab to a large-scale deployment and wider acceptance in the DCN community.

In this work, we present a hybrid DCN testbed setup for DCN scalability and performance studies. This testbed is composed of Polatis free-space Optical Circuit Switch (OCS) [4] with piezoelectric beam-steering, an SDN-enabled DCN simulation platform [5] with System-in-the-Loop (SITL) modules, commercial HP Aruba [6] DCN SDN ToR (Top-of-Rack) switches, high performance network testers (Xena testers [7]) and an OpenDaylight-based (ODL) [8] SDN control framework. We further demonstrate the testbed architecture, explain its large-scale experimental testing capabilities, and present initial performance testing results (latency measurements). The remainder of this paper is organized as follows: section II describes the hybrid testbed setup; section III evaluates flexibility and experimental availability; section IV presents the results and performance analysis, and section V concludes the paper.

II. A Hybrid DCN Testbed Setup

The introduced hybrid (with real physical electronic and optical switches as well as simulated DCN SDN switches) SDN-controlled testbed is presented in Fig. 1 (photo) and Fig. 3. In this scenario, a flat direct-connection Hypercube topology is used in combination with an OCS to form a DCN interconnect. The motivation for assembling and modelling a Hypercube structure, are low latency and high scalability properties, inherent from flattened architectures, widely applied in large-scale HPC (High Performance Computing) clusters. The Proof-of-Concept of building a simpler system with virtual hardware-software linking via SITL was presented in [9] [10].

In this study, an SDN-based control framework is integrated with real-simulated DCN structure, which consists of 2 8-node sub-cubes, namely a physical 8-node cube (HP Aruba 3810M 16-port ToR switches) and a simulated cube of 8 ToR switches in the Riverbed Modeler, together forming a 4-D Hypercube structure of 16 nodes.

The physical part of the testbed consists of a cube, constituting a 3-D cube, consisting of 8 OpenFlow-enabled ToR switches, where each of the ToR switches has a direct physical connection to the Polatis optical circuit switch, as it can be seen in Fig. 3. Server nodes and Xena traffic generators (testers) are connected to the ToR switches, forming a direct-connection structure. The cube may be scaled to any of the variants (e.g.,

[1] This paper was submitted for a review on April 29, 2018. The Authors are with the Technical University of Denmark, Department of Photonics Engineering, Building 343, Oersteds Pl., Kgs. Lyngby 2800, Denmark (e-mail: artpil@fotonik.dtu.dk, srru@fotonik.dtu.dk, ladit@fotonik.dtu.dk). The corresponding author: artpil@fotonik.dtu.dk.

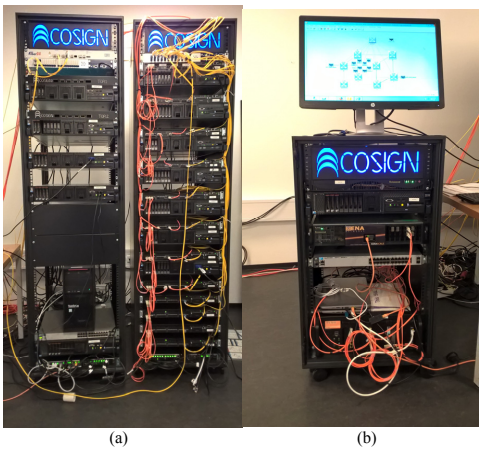
symmetric or asymmetric Hypercube) of an m-dimensional Hypercube subject to the availability of resources, such as OF-(OpenFlow) enabled ToR switches. The intermediate physical implementation is described below.

The physical testbed consists of the following hardware components:

1) 8 x OpenFlow-enabled HP Aruba 3810M SDN ToRs
2) 2 x 48-Port Polatis OCS
3) 12 x 2U server nodes (IBM x3650 M5, IBM x3690 X5)
4) 40 x 850 nm SR (Short Reach) 10 Gbps transceivers
5) 12 x 1310 nm LR (Long Reach) transceivers
6) 1 x XenaBay high performance L2-3 tester
7) 1 x XenaScale high performance L4-7 tester
8) 1 x XenaCompact (1U) high performance portable tester
9) 2 x Control/management plane 24 x 1G Ethernet switches

To generate traffic passing through the hybrid setup, Xena traffic generators are used. These high-performance network testers are providing important set of functionalities, such as:

1) Generation of customizable traffic profiles and packet types for L2-3 and L4-7 network stress-testing; this includes a rich library of pre-recorded communication sessions for the North-South traffic patterns (social networking services, messenger communications, web browsing) as well as more Data Center specific profiles (e.g., point-to-multipoint distributed intra-DC sessions).
2) High accuracy of hardware-based performance measurements (latency, jitter, packet loss, traffic distribution by packet size, inter-frame arrival times, etc.).



(a) (b)

Fig. 1. Rack layout of 4-D Hypercube implementation: (a) 2-rack composition of a physical 8-node Cube with a Polatis OCS, SDN control node (server) and supporting equipment; (b) a setup with simulation framework and Xena network testing equipment.

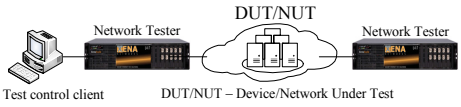


Fig. 2. A common testing approach

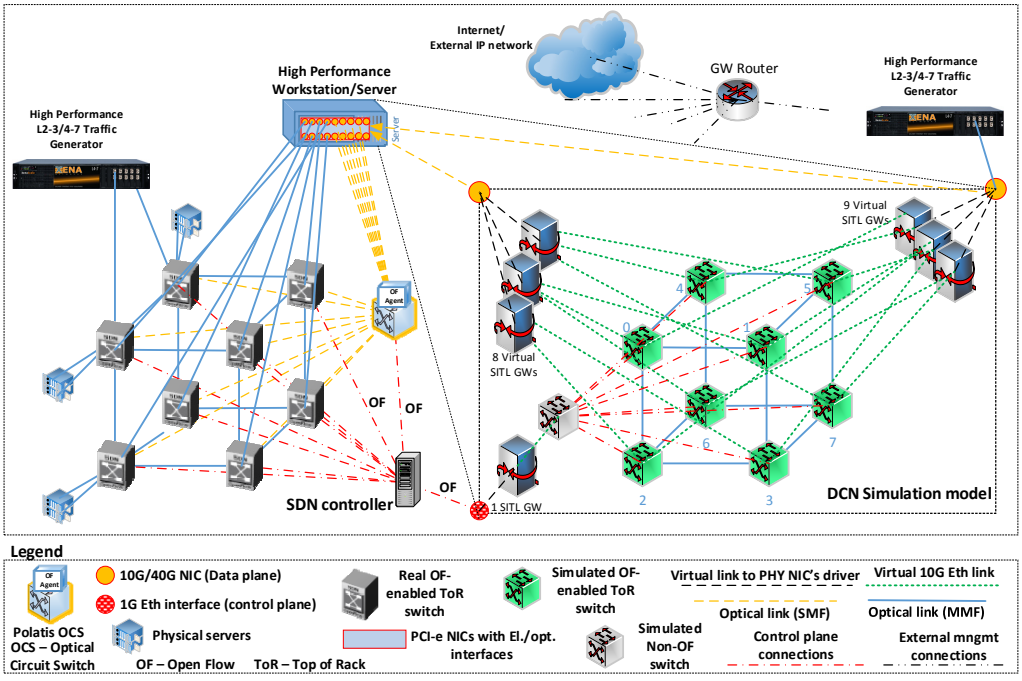


Fig. 3. A hybrid SDN-controlled DCN testbed setup

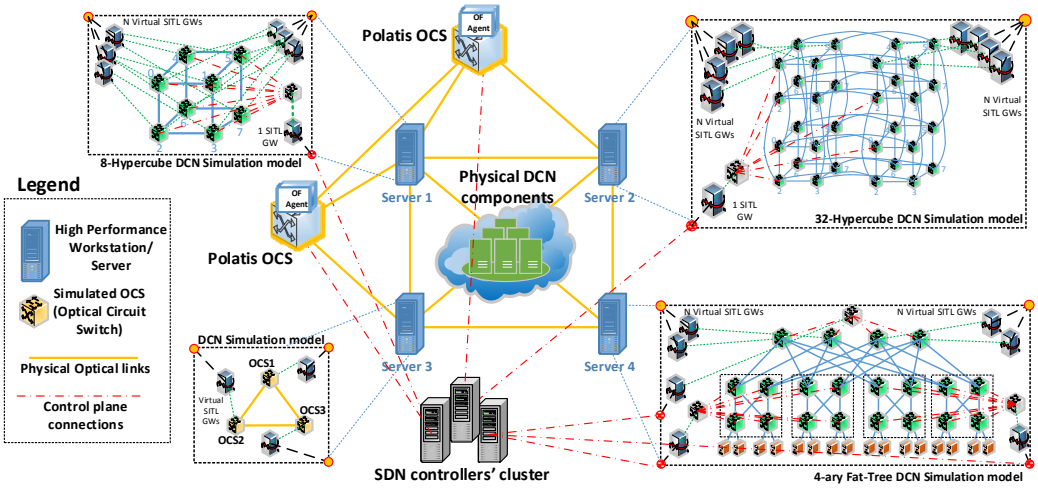


Fig. 4. Experimental flexibility and scalability of hybrid DCN testbed setup

A commonly used network/device testing setup is depicted in Fig. 2, where a test device or network under consideration would be loaded with emulated traffic streams and the statistical information, gathered by the tester, allows understanding the behavior (response) of the device/network in a better-informed and more accurate manner.

There are two types of physical layer connections. First, Multi-Mode Fiber (MMF) links within the physical Hypercube and between the physical ToR switches and simulated ToR switches via electrical-optical (E-O) NICs (Network Interface Cards) of the simulation server. Second, Single-Mode Fiber (SMF) optical shortcut links between physical switches and Polatis 24x24 OCS, and between the simulated switches and OCS.

The simulation environment is hosted by a high-performance server (IBM 2U X3690 X5 with 24 logical cores, 256 GB RAM, 16 E-O interfaces on multiple PCI-e cards). The core of the simulation tool is a real-time kernel and virtualized SITL gateway modules with software-hardware linking capabilities. The simulated ToR switches support two types of logical connections: 1) regular virtual links within the simulated network of ToR nodes; 2) virtual SITL 10 Gbps links. The latter are used to create a logical tunnel from the simulated Ethernet nodes to the driver of the physical E-O port of the NIC, to enable real-time extraction and conversion of the Ethernet frames. High performance Xena network testers are used to enable advanced stress-testing at a DCN scale.

Implementation of a Hypercube-based simulation model consists of network switches, supporting ***OpenFlow 1.3.0 (ONF TS-006)*** protocol [11]. The simulation model is connected to the real physical hardware (network equipment, servers) and an external SDN controller via specialized virtual System-in-the-Loop (SITL) gateways (see Fig. 3.), linked to the physical network interface(s) of a workstation/server, which the simulation environment is running on.

All the network devices (physical and simulated ToR switches, and Polatis OCS) of the hybrid DCN setup are SDN-enabled, creating a flexible, dynamically reconfigurable testbed. The network applications can be configured via REST interface, OSGi framework, as well as using NETCONF (Northbound server) as a Northbound API (NB-API). OpenFlow 1.3+ is used as a Southbound interface (SBI) to interact with all the network devices, as well as NETCONF protocol can now be used to configure and monitor the status of Polatis free-space OCS. The ODL Lithium SR4 controller was customized (added optical extensions to interact with Polatis OCS, interoperability with simulated SDN nodes) to be able to perform a unified control of the real-simulated setup.

The approach of creating a hybrid electrical-optical and physical-simulated SDN-controlled testbed enables more comprehensive and realistic studies of specific DCN communication aspects (both, physical-hardware level and system-level behavior). For example, evaluating repeatable DCN scalability and network protocol effects on a larger scale, or assessment of the optical switching gain (e.g., latency reduction).

The main features and benefits of applying the simulation-based approach to perform the scalability studies are the following. First, the simulation model is configured to operate in real-time, preserving a realistic behaviour and process timings within the simulated SDN-enabled network nodes (switches); this setting is critical to be able to obtain an accurate estimation (real-time sampled measurements) of the packet conversion and buffering latencies at the real-simulated boundary, which will affect the studied network metrics. Second, the support of SDN principles via the implementation of OpenFlow protocol in the simulated nodes offers reasonable flexibility in reconfiguration (the control logic needs to be modified in the SDN controller rather than every individual simulated node), placement of additional components (real or simulated) and increase of the scale of network interconnect.

III. Flexibility and Experimental Scalability

One of the main goals of building such a combined DCN testing system is to achieve scalability property, which can be applied in many performance evaluation scenarios. This is reflected by Fig. 4, showing that the functional scale of the testbed can be further increased by applying parallelization techniques to create a unified cluster of high performance servers to split a complex simulation model of a large-scale DCN into smaller subsets of network elements. The overall traffic processing performance of the system is boosted by using parallel execution mechanism on a multiprocessor system on each physical server of the cluster.

The simulation model of an 8-node (structures of different scale can be modelled) cube is linked to the composed physical DCN testbed in order to analyze the scalability characteristics of this Hypercube-based approach. Therefore, a simulation model is used to scale the DCN to m-dimensional Hypercube ($m > 3$) as well as to evaluate the feasibility of combining physical and software-based network components to build a DCN environment for the research purposes. The following elements of scalability are of particular interest in the context of this setup:

1) Latency/jitter
2) Capacity and flow distribution
3) Effective throughput
4) Energy consumption
5) Network management
6) Hardware performance requirements to support the aforementioned research objectives

The maximum number of hops in Hypercube structures is determined by the dimensionality of the cube, i.e. each node has one, and only one, direct connection to a node in another dimension. This ensures the hop count scales almost linearly with the total number of network nodes. The aim and novelty of the scalability study is not to verify this, but to explore how definable network parameters, such as latency and throughput scale when an optical circuit is available for throughput increase and latency reduction. In order for the results to be measurable, more than 8 network nodes (the content of a 3D Hypercube) are needed, this is where the integration of a simulation model with existing hardware is a reasonable approach.

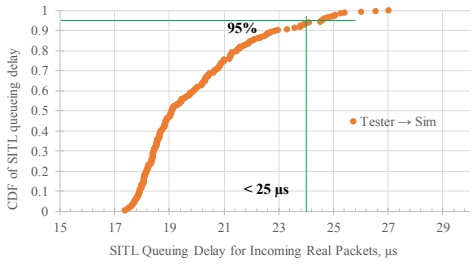
The classical Hypercube structure scales by doubling the number of nodes. This may not be easily realized for a practical implementation when reaching thousands of nodes. For this, an incomplete Hypercube variant is considered. The incomplete variant consists of an asymmetric number of nodes, e.g. 12 instead of 16, thus introducing higher degree of flexibility in scaling the network, instead of just doubling the symmetric structure. The scalability study and the practical steps provided above also include the study of such Hypercubes.

Another application area, as depicted in Fig. 4, is to create a distributed simulation framework to study diverse DCN topologies, enhanced by Software-Defined optical switching capabilities, each controlled by a domain-specific SDN controller from a pool of controllers at the same time. For instance, considering intra- and inter-DCN connectivity scenarios with WAN (Wide Area Network) modelling. This objective can be achieved with the following approach:

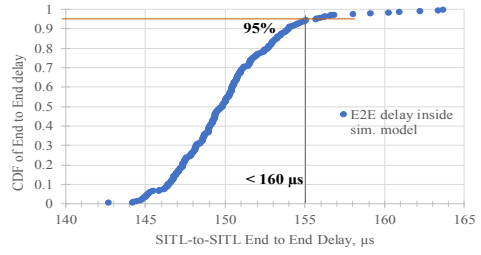
1) ***A single DCN topology***: if only one specific DCN topology is of interest, the simulated part can be better scaled by splitting a large topology into smaller subsets of network nodes and dedicating a physical server machine for each of the network slices. In this way, the processing load can be better distributed over a pool of physical resources, since high packet rate traffic processing in software modules (network nodes of the simulated infrastructure) is a very demanding task and poses a great load on the CPU (Central Processing Unit) subsystem of the server node.
2) ***Multiple DCN topologies***: a heterogeneous testbed structure can be created by modelling distinct DCN topologies on dedicated physical server nodes (a DCN-to-physical-node mapping) and then interconnecting these islands of DCN resources via optical-electrical interfaces and optical links, as conceptually presented in Fig. 4.
3) ***SDN controlled framework***: a great flexibility of the setup is rooted in its high degree of programmability and reconfigurability, since the control plane is fully separated from the forwarding plane due to SDN capabilities, we can implement, enforce and separately test a broad range of network applications and traffic control and processing policies – the main changes need to be performed in the SDN controller, rather than individual DCN components.

IV. Results and Performance Analysis

Performance benchmarking test results, conducted on the setup in Fig. 1, are shown in Fig. 5. It can be seen that 95% of all the real incoming data packets experience a queuing delay



(a)



(b)

Fig. 5. Performance evaluation test results for a hybrid DCN testbed: (a) Queueing delay; (b) Intra-simulation end-to-end delay. Linux CentOS 7 server is used as a host of the simulation platform

(Fig. 5a) lower than 25 µs (95th-percentile) at the SITL gateway interface (real-to-simulated).

On the other hand, internal delay (Fig. 5b) within the simulated DCN infrastructure of 8-Hypercube is, on average, below 160 µs end-to-end (SITL-to-SITL). The system was stress-tested using Xena network testers generating IP (Internet Protocol) traffic flows with incremental rates (but not reaching

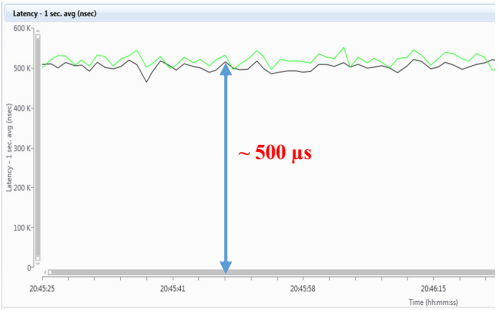


Fig. 6. End-to-end communication latency measured by the Xena Bay tester. Legend: green – aggregate average latency in ns, black – 1 sec. average latency in ns.

the network buffer overflow state in the network switches). The end-to-end traffic flow latency through the entire real-simulated DCN infrastructure (bypassing the OCS in this experiment) was around 500 µs (Fig. 6), considering multiple packet conversion stages.

V. Conclusion

In this paper, we presented a hybrid physical-simulated and SDN-controlled testbed of DCN infrastructure for realistic and comprehensive large-scale performance testing and scalability studies of DCNs with optical switching capabilities. Scalability and application flexibility of the setup are detailed as well. The stress-testing results of the testbed show reasonable packet queueing and intra-simulation processing latencies, under 25 µs and under 160 µs, respectively, allowing to use it in a range of DCN performance studies at larger scale. The hybrid datacenter evaluation framework can be scaled to any topology and to a large scale without abundant expenditures for equipment acquisition.

Acknowledgements

This work has been supported in part by the EC FP7 project "COSIGN, grant no. 619572". We would like to thank Polatis Ltd and Xena Networks ApS for lending the test equipment.

References

[1] M. Imran, et al., "Performance evaluation of hybrid optical switch architecture for data center networks," J. *Opt. Switching and Network.*, vol. *21*, pp. 1–15, Jul.2016.

[2] F. Testa and L. Pavesi, "Optical Switching in Next Generation Data Centers," 1st ed., New York, USA: Springer, 2017, pp. 23 – 44.

[3] J. Aznar, M. Galili, et al., "COSIGN: Combining Optics and SDN In Next-Generation data centre networks," Public Deliverable 5.4, 2017.

[4] Polatis, "Datacenter Networks - All Optical Switching," 2018. [Online]. Available: http://www.polatis.com. Last accessed : 25-04-2018.

[5] Riverbed Technology, "Riverbed Modeler," 2018. [Online]. Available: https://www.riverbed.com/dk/products/steelcentral/steelcentral-riverbed-modeler.html. Last accessed: 25-04-2018.

[6] HP Enterprise, "Aruba 3810M Switch," 2018. [Online]. Available: https://www.hpe.com/us/en/product-catalog/networking/networking-switches/pip.1008605435.html. Last accessed: 25-04-2018.

[7] Xena Networks, "L2-3 (Xena Bay) and L4-7 (Xena Scale) Test Platforms," 2018. [Online]. Available: http://www.xenanetworks.com. Last accessed: 25-04-2018.

[8] OpenDaylight Foundation, "OpenDaylight: Open Source SDN Platform," 2018. [Online]. Available: https://www.opendaylight.org. Last accessed: 25-04-2018.

[9] S. Ruepp *et al*., "Combining Hardware and Simulation for Datacenter Scaling Studies". In *Proc. Of ONDM,* Budapest, Hungary, 2017, pp. 1 – 6.

[10] A. Pilimon and S. Ruepp, "A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks". *Accepted for presentation at ICNC 2018*.

[11] B. Heller, "OpenFlow Switch Specification 1.0.0," 2009. [Online]. Available: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf. Last accessed: 25-04-2018.

Paper G: A Novel Algorithm for Flow-Rule Placement in SDN Switches

A. Mimidis-Kentis*, **A. Pilimon***, J. Soler, M. Berger and S. Ruepp, "A Novel Algorithm for Flow-Rule Placement in SDN Switches", accepted to *4th IEEE Conference on Network Softwarization (NetSoft)*, Montreal, Canada, 25-29 June, 2018, 9 p.

DOI: -

A Novel Algorithm for Flow-Rule Placement in SDN Switches

Angelos Mimidis-Kentis*, Artur Pilimon*, Jose Soler, Michael Berger and Sarah Ruepp
Department of Photonics Engineering
Technical University of Denmark
Lyngby, Denmark
{agmimi, artpil, joss, msbe, srru}@fotonik.dtu.dk

***Abstract*— The forwarding rules, used by the legacy and SDN network devices to perform routing/forwarding decisions, are generally stored in Ternary Content Addressable Memory (TCAM) modules, which offer constant look-up times, but have limited capacity, due to their high capital and operational costs, high power consumption and high silicon footprint. To counter this limitation, some commercial switches offer both, hardware and software flow table implementations, termed hybrid flow table architecture in this paper. The software-based tables are stored in non-TCAM memory modules, which offer higher capacity, but with slower lookup times. In addition, these memory modules are limited in terms of how many requests they can serve per time unit. Thus, exceeding this threshold will lead to packet loss in the network. This paper proposes a novel placement algorithm, which dynamically decides whether a new flow rule should be placed in a hardware (expensive) or a software (cheap) table. The placement decisions are based on a number of criteria with the goal to increase the utilization of the software-based table, without introducing performance degradation in the network in terms of significant delay and packet loss. The performance of the placement algorithm was evaluated through experimental measurements in a testbed, which comprises a hybrid SDN switch, a server performing traffic generation and a server hosting the SDN controller. The results indicate that, by limiting the maximum allowed processing capacity of the software table, the number of accommodated flows is significantly increased, while bounding any excessive delays and avoiding packet loss.**

Keywords— SDN, Flow tables, TCAM, OpenFlow Pipelines

I. INTRODUCTION

Regardless of the applied control plane paradigm (centralized or distributed), network devices make routing decisions based on rules that reside within one or more device-local rule tables. Whenever a packet arrives in a network/forwarding device, its headers are cross-checked against the rules that populate those tables and depending on the result (match or no match) one or more actions might be applied to the packet by the device (e.g. forward, drop). The granularity with which the packets are examined (i.e. the number of headers checked), and the implementation and/or number of tables involved, can vary between different devices and networking paradigms.

Given the need for fast packet processing, most physical network devices implement their forwarding tables using Ternary Content Addressable Memory (TCAM) modules, which provide O(1) lookup times (in terms of clock cycles) [1]. That means that regardless of the number of entries in the table(s), finding a match (or not) will always take the same amount of time. This trait is highly desirable; since apart from generally fast search times, it also provides a high level of determinism (all look-ups take always the same time). However, TCAM modules are expensive, have high power consumption and a large silicon footprint [1]. In order to cut-down on the associated Capital and Operational Expenditures (CAPEX, OPEX), network device vendors limit the size of the TCAM modules, which results in reduced number of flow-rules to be stored [1].

The Software Defined Networking (SDN) paradigm offers a greater granularity for defining flow-rules (more packet headers can be defined as match fields), when compared to the traditional network paradigm. This approach provides increased flexibility when designing applications for SDN Controllers (SDNCs), but requires more memory space per flow rule. Given the limited size of the flow tables, this characteristic can limit the applicability of SDN in network deployments with extensive number of flows. To resolve this issue, both SDN-enabled device vendors and SDNC application developers (from the industry and the academia) have attempted to increase the effective capacity of the flow tables. With regards to vendors the approach is to provide an additional flow table implementation in their devices based on software. This can offer increased flow-rule capacity by sacrificing search performance (exceeding O(1)). On the other hand, SDNC developers have mostly focused on developing flow-rule aggregation mechanisms, which allow the network devices to process more traffic with the same number of flow-rules. The drawback of this approach is that aggregation of network flows reduces the processing granularity offered by SDN. These approaches are covered in more detail in the related work section of this paper.

This work builds on the software table implementations, by proposing a dynamic and intelligent flow-rule placement algorithm, executed in the SDNC. The target is to maximize the number of flow-rules residing in a switch, while also

978-1-5386-4633-5/18/$31.00 ©2018 IEEE

limiting the negative effects (e.g. increased delay, packet loss), imposed by the memory modules (hardware or software).

The remainder of this paper is structured as follows. Section II provides an overview of the related research work based on flow-rule placement and flow aggregation algorithms. Section III discusses the available switch architectures, focusing on their flow table implementations (pure hardware, pure software and hybrid). Section III discusses flow table pipeline implementations and how they can affect the performance of flow-rule placement algorithms. Section IV presents the proposed flow placement algorithm for hybrid flow table architectures. Section V describes the experiments conducted to evaluate the proposed algorithm, together with the collected results. This paper is concluded and possible future steps are outlined in Section VI.

II. Related Work

Efficiency of the flow rule installation process in SDN switches has been a matter of high research interest in the SDN community. Different approaches to manage the TCAM space utilization have been proposed and evaluated, primarily targeting the flow rule compression or aggregation [2 – 5], or flow rule caching and placement algorithms [6 – 12]. Hence, we further provide a summarized yet comprehensive overview of the relevant literature findings.

A. Flow rule aggregation

Flow rule aggregation aims at reducing the flow table size in the network nodes by substituting a set of rules with overlapping matching criteria with a more generalized flow rule, while still being able to realize a corresponding network policy. The variants of this procedure are commonly referred to as traffic flow aggregation [2][3] and flow table compression [4][5]. An approach for dynamic flow aggregation was proposed in [2], where a dynamic traffic aggregation decision is made based on two criteria: the computed network path of the flows and their DSCP (Differentiated Services Code Point) marks. The traffic flow aggregates are identified by adding unique per-flow-aggregate VLAN (Virtual Local Area Network) tags. However, the performance of the aggregation service was measured in a simulated network with software switches (on a single physical server) and a limited number of traffic flows. Rifai *et* al. in [4] proposed a framework, called MINNIE, for flow table compression using wildcard rules and shortest-path routing using adaptive metrics (link, router load) for load balancing. The compression mechanism produces a set of three tables, using compression by source, by destination and by default flow rule. The smallest resulting compressed table is chosen for the routing decisions. Experimental measurements, described in [4], were conducted , on a testbed, containing a commercial SDN switch with a hybrid (software-hardware) flow table design. The results show that even when using this compression technique, the first packet (of each flow) delay increases by a factor of up to 20 and the average matching delay for the remaining packets results in 6-fold increase when installing the flow rules in software as compared to hardware (TCAM). An incremental flow table aggregation mechanism is discussed in [5], where the authors propose a set of two algorithms, namely FFTA (Fast Flow Table Aggregation) scheme applied to non-prefix (TCAM-based) flow rules. An offline version of FFTA is used for initial partitioning of the flow rules, applying prefix aggregation and then merging together the rules with a single differing bit in an iterative manner. The online version allows performing fast incremental rule updates with a small loss of compression ratio.

B. Flow rule placement algorithms

This category of flow rule distribution methods includes flow rule caching and rule placement in general. In [6][7] authors present a design of a hybrid hardware-software switch abstraction with arbitrarily large flow rule tables. This is achieved by using a complex rule caching mechanism, consisting of a rule placement algorithm, called CacheFlow, a Cache master module and a set of elastic shared software switches. The rule placement algorithm constructs a rule dependency tree and caches the most popular flow rules (serving a large volume of "cache hit" traffic) in the TCAM, but redirects smaller amount of "cache miss" traffic to be handled by the software switches. If there is no matching rule found either, the controller is contacted as the last instance for a new flow rule installation. This system allows achieving several important goals: a) avoid cache replacement without taking into consideration possible complex flow rule dependencies (pattern overlaps); b) avoid flow compression to preserve the OpenFlow semantics, i.e., per-flow-rule traffic counts; c) reduce the size of the long chains of dependent rules by "splicing" such chains to cache smaller groups of rules [7]. Another flow rule caching optimization method, called CRAFT is introduced in [8]. This mechanism uses a two-stage pipeline to eliminate the need for slow processing of long rule dependency chains, to reduce the possibility of having overlapping flow rules in the space-limited cache. The cache expansion problem is solved by weighted splitting of large flow rules into sub-rules and only caching the sub-rules with the highest weight (hit ratio). This scheme is reported to be 30% more efficient as compared to the CacheFlow [7].

Guo *et* al. in [9] propose a novel traffic forwarding scheme coupled with reactive flow rule placement, called JumpFlow. The forwarding module of the algorithm uses the VLAN identifier (VID) field of the packet header to carry the routing information, while the rule placement module divides the complete flow's forwarding information into several blocks and installs them on a selected subset of contact switches (along the flow's path). The objective of the reactive module is to maintain low and balanced flow table (TCAM) utilization by applying constraints of flow table space and the number of contact switches to use, with an optimal solution achieved using Integer Linear Programming (ILP).

In [10], the authors employ a flow rule partition and allocation strategy, where the flow rules in heterogeneous flow tables are split and grouped into sub-tables (stored in a virtual small TCAM block), which are then distributed across the entire network as uniformly as possible. Only the hardware (TCAM) flow tables are targeted. The main goal of this

approach is to divide all flow rules into disjoint sub-tables, putting the rules that implement the same policy or have dependency in the same sub-tables.

A novel solution to optimize the TCAM memory usage is proposed in [11], by implementing a Memory Management System (MMS) component for the SDNC. It performs memory swapping by temporarily moving the least used flow rules from the TCAM space to the external database (residing in the MMS of the SDNC). Then, when the load of the TCAM table decreases, the MMS automatically restores the swapped-out rules upon demand (e.g., a new packet matching one of these rules arrives).

Other solutions, e.g., as in [12], focus on flow-driven rule caching optimization, where authors achieve a high cache hit ratio by prefetching (caching over all the switches along a flow's path) the flow rules that need to be cached for fast path processing and setting a timer with an estimated time of the next rule "hit" event. This is achieved by analyzing the routing paths of each flow and its detected traffic pattern.

Our proposed rule placement approach differs from related work in several ways, even though some conceptual similarities with the discussed works are present. First, we are not targeting the flow table compression to retain the possibility to obtain per-flow-rule traffic statistics and avoid introducing any need for recalculating the optimal number of compressed rules, which can be an NP-hard task, since we are considering a dynamic reactive flow rule migration. Second, unlike in the case of a CacheFlow [6][7] approach, where additional processing overhead is introduced by embedding the software switches (with additional pipeline processing) and extra coordination component (cache master), we utilize the properties of the hardware and software tables and keep the main algorithmic logic in the SDNC. Third, we are not modifying the packet headers to perform flow grouping by similar properties (e. g., packet rates), but instead we are using a predefined mapping of flow group rates to transport protocol destination ports. Finally, our work is conceptually related to [11], since we are swapping the flow rules between different memory types, but we retain this process within the memory space of the switch, rather than exporting the rules externally that incurs varying delays.

III. SWITCH ARCHITECTURES

When evaluating the performance and utilization of flow table implementations, there are three architectural components that must be taken into consideration. These are:

- How the flow tables are implemented (pure hardware, pure software or hybrid)
- How the flow tables within a single device are interconnected; a mechanism referred to as the *packet processing pipeline* of the device.
- How flow rules are allocated between the different flow tables; a mechanism referred to as a *flow rule placement algorithm*.

A. Flow table Architectures

There are three means to implement flow table architectures. They can be realized using pure hardware resources (e.g. TCAM), pure software resources or in a hybrid combination of these two, where some tables are implemented in hardware and some in software.

Pure software flow table implementations are almost exclusively encountered in virtual switches (e.g. OpenvSwitch [13]). Virtual switches are mostly used in Data Center (DC) environments, to forward traffic between virtual machines or containers, which reside within a single physical node. Because of their locality these switches handle only a limited number of traffic flows, hence the associated look-up operations do not impose significant performance degradation. Pure hardware implementations are mostly found in physical, legacy (non-SDN) network devices. Hybrid implementations are a relatively new approach, most commonly found in SDN-enabled network devices. This is because, through the programmability offered by the SDNC, dynamic flow rule placement algorithms can be implemented and enforced in the network infrastructure.

As summarized in Table 1, each flow table implementation has several benefits and drawbacks. Pure hardware implementations offer a fast (and constant) per packet look-up and also a high processing capacity, meaning they can handle traffic of high packet rates. However, their flow table capacity is limited, due to the CAPEX and OPEX costs associated with the TCAM modules. Pure software implementations on the other hand, offer a much higher flow table capacity, but at the cost of slow look-up and low processing capacity. Since they do not require a flow table placement algorithm, both hardware and software implementations have a relatively low complexity. Finally, hybrid implementations (if correctly utilized) can offer high flow table and processing capacities, as well as fast look-ups. The only disadvantage is the need for a placement algorithm, which can increase implementation complexity. However, in this paper we argue that if the placement algorithm is of a simple and efficient design, the benefits can out-weight the introduced complexity.

Table 1: Comparison of flow table implementations

Type of Implementation	Flow table Capacity	Lookup Speed	Processing Capacity	Complexity
Pure Hardware	low	high	high	low
Pure Software	high	low	low	low
Hybrid	high	high	high	high

B. Packet Processing Pipelines

Within the context of the SDN paradigm, a packet processing pipeline refers to the logic of the internal packet processing within a network device. There are two approaches for designing flow table pipelines, using a single flow table in which to store all flow rules or using multiple interconnected tables and store rules in them based on a set of criteria. The approach is dependent on both the underlying capabilities of the network device, but also on the protocol used for the control plane between the SDNC and the device (e.g.

OpenFlow 1.0 does not support multi-table pipelines but OpenFlow 1.3 does). When considering a single table pipeline, all incoming packets in the network device are cross-checked against this table. In case there is a match, the packet is processed based on the actions associated with the matching rule; if there is no match then the packet is sent to the SDNC. In a multi-table pipeline, flow processing can be composed of multiple flow rules, spread across the different tables. This means that an incoming packet can be processed by multiple tables, allowing for more complex action sets to be enforced. Using a single flow table offers a lower implementation complexity, but using multiple flow tables allows for more efficient and dynamic flow table utilization. Figure 1 illustrates the two pipeline approaches.

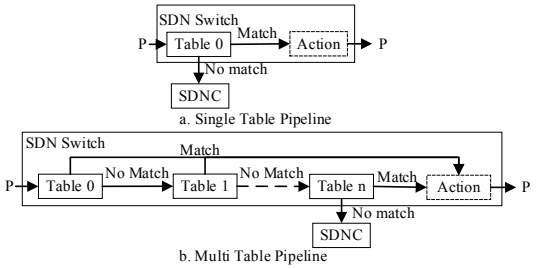


Figure 1: Pipeline models

IV. Proposed Pipeline and Placement Algorithm

As mentioned, hardware flow tables offer high and constant service rates (packets per second they can process), but are limited in the number of flow rules they can accommodate. On the other hand, software flow tables can accommodate more flow rules but have limited service rates. Additionally, for software tables the time it takes to service a request is directly related to the current number of flow rules in the table, which implies that their performance deteriorates as the number of flow rules, present in the software table, increases. This section presents the design and logic of the proposed flow rule placement algorithm and the selected packet processing pipeline. Both the placement algorithm and the pipeline, were implemented with the aforementioned benefits and drawbacks in mind.

A. Packet Processing Pipeline

In the SDN paradigm and with OpenFlow [14] as the control plane protocol, when a switch receives a packet, it cross-checks it against its flow rules for a match and then applies the associated actions to it. The outcome, however, is dependent not only on the defined action set, but also on how the pipeline processing within the switch is implemented. In this work, it was decided to process the incoming packets first at the hardware table and then the software table. This approach removes unnecessary processing stress from the software table as it is only accessed when a match is not found in the hardware table. If neither the hardware of software table holds a matching flow rule, then the switch will ask the SDNC for further instructions with an OpenFlow PacketIn message. Upon receiving the PacketIn message, the SDNC will process it and decide how the packet should be treated in the network (f. ex. forwarded, dropped, modified etc). The means through which, the SDNC processes the request and decides on the packet treatment is out of the scope of this paper. After the packet treatment has been decided by the SDNC and before it is enforced in the network (by means of OpenFlow FlowMod messages), the proposed flow rule placement algorithm takes place. The implemented pipeline is illustrated in Figure 2.

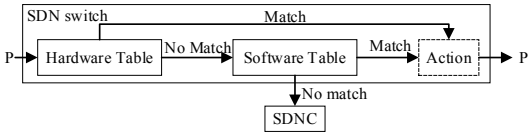


Figure 2: Proposed pipeline processing

B. Flow Rule Placement Algorithm

Upon receiving a request from a switch, the SDNC will query a local statistics database and retrieve the number of flow entries that populate the switch's hardware flow table. The information contained in the database is collected by means of a polling mechanism, which periodically retrieves switch related statistics. Even though the SDNC is the only entity managing the network, it is not safe to assume that it can keep track of the number of active flow entries in each switch without such a polling mechanism. This is because some of the flow rules in the switches might expire and be removed without the SDNC being notified (e.g., the SDN switch might not send an OpenFlow FlowRemoved message to the SDNC). By setting a high polling frequency, the accuracy of the collected statistics can be set to acceptable standards, at the expense of extra overhead in the interface between the SDNC and the network infrastructure.

Upon retrieving the number of flow rules in the hardware table, the algorithm compares the value against a predefined threshold. If the number of flow rules is below this threshold, then there is no imminent danger of overflowing the hardware table. Since the performance of the hardware table is superior to the software table, the flow rule is added to the hardware table. If the number of flow rules is greater than the defined threshold, then there is a danger that inserting the flow rule in the hardware table will cause a table overflow and disrupt network connectivity (e.g., packet drops, SDN switch crash). To mitigate this danger, the placement algorithm triggers a flow rule migration process from the hardware table to the software table. There are two elements of this migration process that need to be addressed here, namely how many flow rules are migrated each time the process is triggered, and which flows are selected for migration. Based on the issues addressed above, Figure 3 illustrates the proposed algorithm in the form of a flow chart. Table 2 lists the different variables used in the chart.

With regards to how many flows to migrate, three approaches have been identified. The algorithm could migrate one flow, migrate K flows (K could be either a static predefined value or dynamic based on the current situation), or

finally the algorithm could migrate as many flows as possible until the service rate of the software table is saturated. Each approach has its own benefits and drawbacks which are presented below.

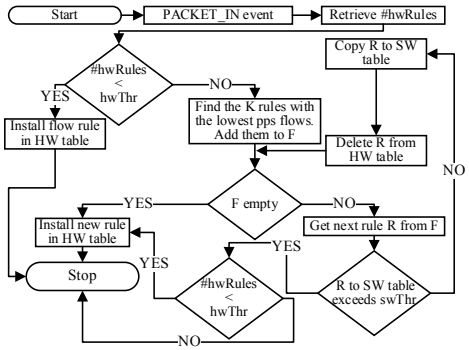


Figure 3: Flow chart of the flow rule placement algorithm

Table 2: List of flow chart variables

Variable	Description
#hwRules	The number of flow rules existing in the hardware table of the switch.
hwThreshold	The threshold, expressed as number of flow rules, which identifies a critical point after which the hardware table is prone to table overflow.
swThreshold	The threshold, expressed as packets per second, which identifies a critical point after which the software table can become unresponsive.
F	A list which holds all flow rules considered for migration.
R	A single flow rule considered for migration
K	The static or dynamic value, denoting how many flow rules to consider for migration on every iteration.
pps	Packet per second rate of a flow.

Migrating as many flow rules as possible, reduces the instances in which the threshold is reached, hence limiting the number of times the migration process is initiated. However, this approach always leads to the full utilization of the software table, which will hinder network performance due to increased delays for the migrated flows. In contrast, migrating just one flow rule whenever the algorithm is triggered minimizes the utilization of the software table. However, unless some flow rules from the hardware table expire or are removed by the SDNC, this approach requires one iteration of the placement algorithm for each new flow arriving at the switch. This makes the algorithm more computationally expensive, as well as it increases the response time of the SDNC to service requests from the network. The final approach and the one selected for this work, is to migrate K flow rules per iteration of the flow rule placement algorithm. Doing so provides the benefits of both previous scenarios, since the number of times the algorithm is triggered is limited but so is the utilization of the software table. It is important to stress that independent of the selected approach (1 flow rule, K flow rules, max flow rules), a flow rule should be migrated to the software table if and only if, the resulting cumulative packet per second rate of the software table is under a predefined threshold. Exceeding this threshold means exceeding the processing capabilities of the software table, leading likely to disruptions in network connectivity. If that is the case and the hardware table can accommodate the flow, it will be added there. Else the flow will be dropped (neither the hardware nor the software table can accommodate it).

Based on the correlation between service requests (packets per second) and service times (time it takes to find a matching flow rule) for software tables, the proposed algorithm selects the K flows with the lowest packet rate for migration. This way the utilization of the software table is kept to a minimum. Most SDNCs, offer flow level statistics which include per flow packet rates, however, the accuracy of these statistics is very coarse as they are based on periodic polling with an interval at the order of seconds. Given that the packet rate of a flow can vary significantly during its lifetime, using these statistics can lead to incorrect assumptions on the flow's packet rate. Migrating a flow rule based on a wrongly assumed packet rate can lead to over provisioning of the software table which, in turn, can lead to either packet losses or excessive delays. The means, through which the packet rates of flows are identified, are out of the scope of this paper. However, some possible solutions are either the increase of the polling frequency from the SDNC to sub-second values or the use of network analytics techniques (e.g., sFlow). For the proof of concept implementation of the algorithm, the packet rates of each flow are considered constant and are also identifiable from the SDNC by packet header values, where each destination UDP port implies a specific packet rate.

To avoid network connectivity disruptions for the flows of the migrated flow rules, a migrate-then-delete approach was selected. Since the hardware table resides first in the pipeline processing, this model assures that there will always be at least one active copy of the flow rule within the switch. The drawback of this approach is that temporarily there will be two identical flow rules in the switch, one on each flow table. However, this only holds true for a very limited amount of time, since the migration process is executed relatively fast. Figure 4 illustrates an example of a flow rule migration instance for K = 3.

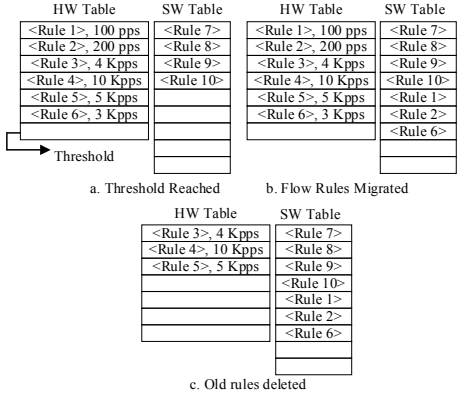


Figure 4: Example of the flow rule migration process, with K = 3

The flow rules 1, 2 and 6 are migrated since their corresponding flows have the lowest packet rates. In (a), the SDNC identifies that the threshold for the hardware table has been reached so it will initiate the migration process. In (b) the flow rules 1, 2 and 6 are migrated to the software table and in (c) they are deleted from the hardware table.

The last element that needs to be addressed is how the threshold values for the hardware (number of rules) and software (packets per second) tables are set by the algorithm. Defining the threshold for the software table is straightforward, since the packet service capacity of the software table is known from the device's datasheet and is independent of any variables (e.g. packet size). Defining a threshold for the hardware table on the other hand is a more complex task. This is because the number of rules that a hardware table can accommodate is not a static value but can vary depending on how coarse/granular each installed flow rule is. The more header fields are defined for matching in a flow rule, the more space this flow rule occupies in the table. Based on this observation, there are two approaches that can provide a secure threshold value. The first is to calculate exactly how many bytes each flow rule occupies and then sum all the values together; the sum can then be compared against the total space provided by the hardware table. However, this approach implies knowledge of how much space each unique header field will occupy, information not necessarily available to the SDNC. Another approach, and the one selected for the PoC implementation, is to follow a worst-case scenario in which it is assumed that all flow entries occupy the same amount of space, equal to the case in which all header fields are defined for matching. This approach has the obvious drawback of limiting the effective size of the hardware table, but due to its simplicity it was selected for the PoC. As a future step, a more robust mechanism for calculating the available space should be implemented.

V. Validation and Results

To validate the functionality of the implemented PoC algorithm, a set of experiments was conducted on a physical SDN testbed. The testbed comprised a server for generating and receiving traffic flows, a physical SDN switch with hybrid flow table implementation and finally a server hosting the SDNC [15] in which the flow rule placement algorithm was executing. The server, responsible for generating the traffic flows, was equipped with 2-port NIC with one port for transmitting and one for receiving the traffic flows. Both NIC's ports were then connected to the SDN switch. The reason for using a single server for sending and receiving traffic flows was the need for a common reference clock for the delay measurements. Finally, the SDN switch was connected to the SDNC though the management interface. Figure 5 illustrates the testbed setup that was used.

The scope of the presented experiments is threefold. First, to validate that the flow rule placement algorithm works as intended by migrating flow rules from the hardware to the software table, based on the defined threshold values. Second, to evaluate, if the algorithm introduces any performance degradation in the network, when compared to the default scenario, where all flows are placed in the hardware table. Third, to observe the combined impact of higher flow pps rates when migration is activated, while keeping the packet loss as low as possible to ensure accurate latency measurements. Due to the limitations imposed by the traffic generation software, it was not possible to saturate the capacities of the hardware and software tables. To mitigate this issue, the SDNC was utilized to "virtually" cap the capacities of both tables to lower values. For the hardware table the limit was set to 99 flow rules and for the software table to either 400 or 600 packets per second (pps) depending on the experiment, defined as follows.

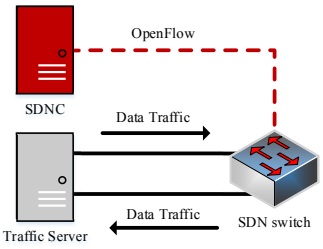


Figure 5: Testbed

The following traffic generation experiments were designed with the scope of stressing the (capped) capacities of both the hardware and software tables. There are two experiments with 150 unique traffic flows in each, with the flows evenly spread amongst three packet rate groups. In the first scenario, there are 50 flows with 10 pps, 50 with 20 pps and 50 with 30 pps. The second scenario is comprised of 50 flows with 15 pps, 50 with 30 pps and 50 with 45 pps. The capacity of the software table was limited to the 400 pps for the first scenario and to 600 pps for the second scenario, with the intent to be able to reach the overflow state for the software table in both scenarios. The traffic flows were sequentially generated in a round robin fashion from each packet rate group within each scenario. Execution of the experiments resulted in the expected behavior. Initially, all flow rules were installed in the hardware table, however when the hardware threshold was reached (set as 95% of the capacity), then the migration process was initiated, and a set of flows to migrate from the hardware to the software table was iteratively being chosen. This process was repeated until the processing capacity of the software table was saturated, and the migration process stopped. After this point the hardware table utilization reached 100% of capacity, and all subsequent flows were rejected.

To evaluate the performance of the algorithm the same experiments as before were repeated with and without the placement algorithm enabled. In the first set of experiments, which will be referred to as *baseline scenario*, only the hardware table is used to serve the arriving flow processing requests. The second set of experiments will be referred to as *flow migration scenario* and are used to benchmark the

performance (in terms of delay) of the flow rule placement algorithm.

It is important to note that the timescales (on the horizontal axes) of the graphs, presented further, are relative per-flow timescales, rather than on a single universal timescale for all the flows. Hence, the last plotted value on any per-flow timescale denotes the total flow duration in seconds. However, this aspect does not affect our analysis, since we are not plotting the delays of groups of flows in a single graph as a function of time.

The results for the *baseline scenario* are presented in Figure 6 and Figure 7. As it can be seen in Figure 6, the distribution of the average per-flow delay for both packet rate sets (10-20-30 pps and 15-30-45 pps) in the baseline scenario is very similar to a uniform pattern with the mean value around 0.175 ms for the first set, 0.179 ms for the second set, and the average maximum delay not exceeding 0.2 ms. Such performance is expected, because the flow rules are placed only in the hardware table (which offers constant lookup times); if there is no remaining space to accommodate a new flow, the packets of that flow will be dropped. This is the behavior illustrated in Figure 6, where there are 99 accommodated flows (out of 150), adhering to the virtually imposed hardware table capacity limit (99 flow rules).

In addition, as it can be seen in Figure 6, there is a tendency of a near-linear latency increase within each packet rate group (of both sets) with the increase of the number of accommodated flows. This can be attributed to the increase of traffic load over time. Figure 7 shows the per-packet delay distribution of a sample flow from the first set of group rates with the mean (μ) and standard deviation (σ_D) values. The results show that per packet delay variation (with $\sigma_D = \pm 0.079$ ms) of a flow, served by the hardware table, is not experiencing significant fluctuations over time and remains relatively stable. This is a behavior that meets the expectations of a flow installed in the hardware table.

With regards to the *flow migration scenario*, the distributions of the average per-flow delays of the flows from the first set of packet rate groups (10-20-30 pps) for both scenarios (baseline and migration) are compared in Figure 8. There are 44 migrated flows that now experience higher average delays, since they are served (for a portion of their lifecycle) by the software table, as compared to the other flows, which were not affected by the migration process and were served only in hardware. In this experimental setting with predefined parameters (e.g., the total number of flows defined, the number of flows to consider for migration in each iteration, the capacity thresholds of the flow tables), the migrated flows belong only to the lowest 10 pps group since the accumulated pps rate of the group (500 pps) exceeded the software threshold limit (400 pps). For the remaining flows the impact is similar to the baseline scenario. This indicates that the placement algorithm does not affect the performance of the non-migrated flows. This is also confirmed in Figure 9, where the per-packet latency distribution of a sample (non-migrated) flow does not change significantly over time and is comparable to the baseline case in Figure 7.

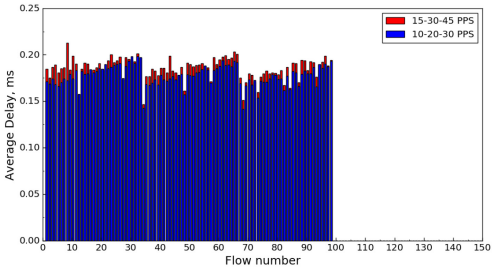


Figure 6: Baseline scenario. Average per-flow delay for 2 rate group sets

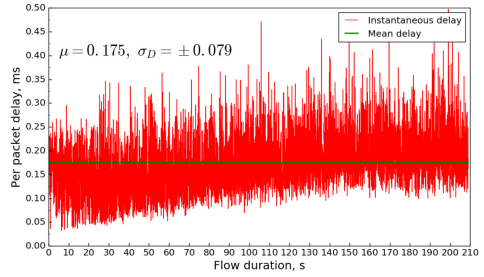


Figure 7: Baseline scenario. Per-packet delay of a sample flow

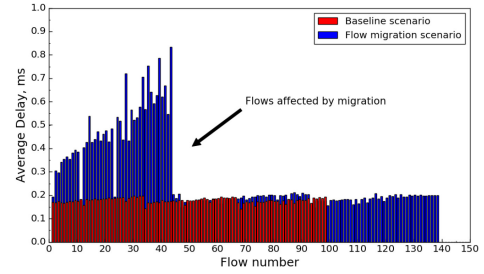


Figure 8: Baseline vs Flow migration scenario. Rate group set: 10-20-30 pps

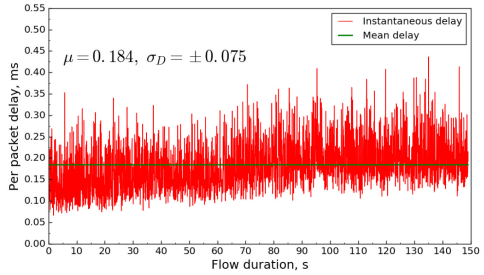


Figure 9: Per-packet delay of a sample non-migrated flow (migration enabled)

Considering the total number of accommodated flows, it is increased to 138 when migration is enabled, as compared to 99 in the *baseline scenario*, while the remaining flows were rejected as expected, because the capacities of both flow tables were fully utilized.

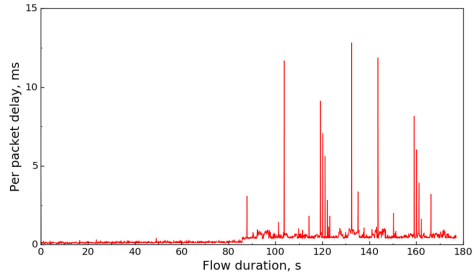


Figure 10: Per-packet delay of a migrated flow (10-20-30 pps set)

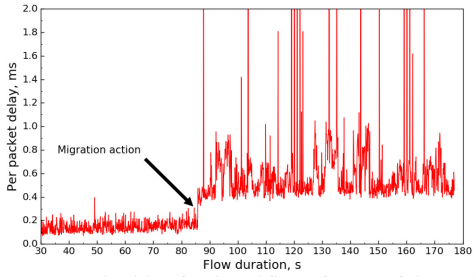


Figure 11: Per-packet delay of a migrated flow (a fragment of Figure 10)

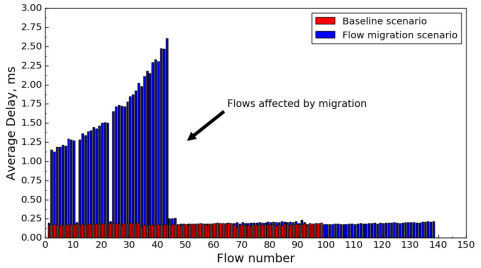


Figure 12: Baseline vs Flow migration scenario. Rate group set: 15-30-45 pps

Another aspect is how the packet processing delay changes when a flow rule is migrated. Figure 10 presents the evolution of per-packet processing delay of a sample migrated flow, and Figure 11 shows a zoomed-in fragment of it. We can observe a sharp increase of latency (the migration point in Figure 11) after the migration process is completed, since the processing is handled in the software table from there on. The impact of software processing is clearly seen in the form of stochastic latency spikes that can be a result of having shared interrupt-based processing in the CPU (Central Processing Unit) and memory buffer resources of the switch. The delay evolution pattern of all the migrated flows of this set of group rates is identical, with a sharp latency jump, higher delay variance and spikes after the migration.

The per-flow delay measurement results for the flows from the second set of packet rate groups (15-30-45 pps) for both scenarios (baseline and flow migration) are depicted in Figure 12. The distribution of the average delay of the non-migrated flows has identical pattern as compared to the baseline case. The increase of per-flow-group packet rates by ~ 33.33% resulted in a corresponding threefold increase of the average per-flow delays, and the increase pattern is observed to be non-linear.

The latency evolution of the individual migrated flows from this rate group set indicates that there is a significantly larger density of latency spikes with an area of excessive high magnitude spikes, reaching up to 100 ms. This behavior is presented in Figure 13 and its enlarged fragment in Figure 14. Such packet processing effects were observed in all the migrated flows and appeared at relatively the same (universal) points in time; these results indicate that larger number of packets are experiencing performance degradation due to higher load on the CPU-based subsystem of the switch.

It is important to emphasize that the observed spikes in delay appear after all the flows have been installed in the switch by the SDNC (in both the hardware and software tables), at which point the SDNC was not issuing any flow-rule-related actions in the switch. Thus, this behavior is purely associated with the switch, and not with the SDNC and/or the implemented placement algorithm.

The observed general variation of the measured delay, present in both scenarios, can be a consequence of the inherent hardware processing effects, e.g., clock drift and clock skew of the traffic generation server, affecting the packet timestamping accuracy, and internal memory buffer limits as well as packet queueing delays in the SDN switch.

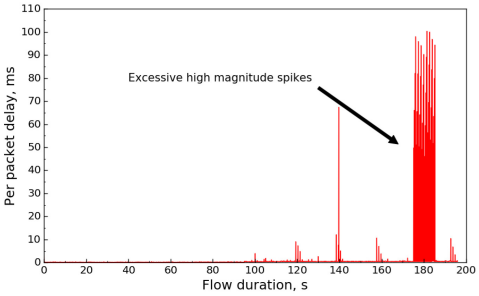


Figure 13: Per-packet delay of a migrated flow (15-30-45 pps set)

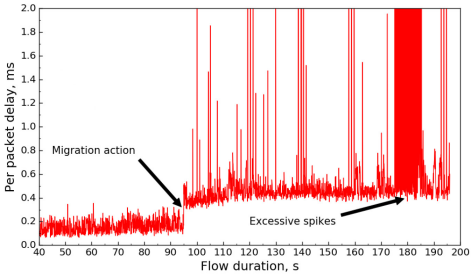


Figure 14: Per-packet delay of a migrated flow (a fragment of Figure 13)

As illustrated in Figure 8 and Figure 12, there is a trend of a sequential increase in average delays for the migrated flows, as more and more flows are accommodated. During the

migration process the flow rules, to be migrated, are retrieved by means of SDNC-specific functionality, without the opportunity to order them in a custom way. Assuming that the last flow installed in the hardware table of the switch is the first flow retrieved from the list provided by the SDNC, the trend shows that the flows that spent most of the time in the hardware table are experiencing the lowest average delays. This is a behavior that follows the performance characteristics of the two table implementations. Finally, we compared the distributions of the per-flow packet loss in both scenarios, and the results show that no packet loss was experienced.

It is important to emphasize that, since we had to virtually cap the processing capacity limits of the hardware and software tables, due to the limitations of our testbed setup (traffic server), we were not able to reach the effective (maximum) processing capacity limits of these tables. Therefore, if the real processing limits would be reached, the results could evolve in a different (non-linear) way, and the performance trends presented in this work, would have to be adjusted accordingly. However, even with virtual capacities, a clear trend was observed in the performance characteristics of the software table implementation. To obtain more indicative results, we need a more accurate traffic generation and measurement mechanism to be able to find the optimal values of the table performance settings, e.g., DPDK-based (Data Plane Development Kit) [16] solution.

VI. Conclussions

This work presented a flow-rule placement algorithm for SDN switches with hybrid flow table implementations. The algorithm is designed to utilize the flow rule capacities of both hardware and software tables, whilst also taking into account their inherent characteristics and limitations. The algorithm was implemented for the ONOS SDN controller and validated/evaluated on a physical SDN testbed. The results indicate that using the placement algorithm allows accommodating a larger number of flows, while limiting the degradation in network performance for the migrated flows and without impacting the non-migrated flows. Apart from that, the algorithm does not incur any packet loss. The downside is stochastic delay spikes affecting the migrated flows, which are caused by the inherent limitations of software-based processing of the switch. Since the behavior of the software table heavily depends on the flow packet rates we believe that the use-case of the algorithm could be to offload low pps low priority flows to the software table. However, for the algorithm to be able to perform, the switches, on which it is going to be enforced, must first be evaluated in terms of their software table performance, so that the appropriate thresholds can be set. Finally, a set of future work proposals is outlined as follows: 1) It might be of interest to model the performance of the software table, with regards to its utilization. The results can then be used as feedback on the placement decisions; 2) for a non-PoC implementation of the algorithm, the per-flow packet rates should be measured using a dynamic and accurate channel (e.g. sFlow); 3) more accurate traffic generation and measurement means should be used to be able to perform stress-testing of the SDN devices and the developed algorithm.

Acknowledgment

This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programmes, under Grant Agreement No. 761 557 (http://ngpaas.eu).

References

[1] K. Pagiamtzis, S. Member, A. Sheikholeslami, and S. Member, “Content-Addressable Memory (CAM) Circuits and Architectures : A Tutorial and Survey,” vol. 41, no. 3, pp. 712-727, Feb. 2006.

[2] A. Mimidis, C. Caba, and J. Soler, “Dynamic aggregation of traffic flows in SDN: Applied to backhaul networks,” In *Proc. IEEE NetSoft Conf. Work. Software-Defined Infrastruct. Networks, Clouds, IoT Serv.*, Seoul, pp. 136-140, 2016.

[3] S. Das *et al.*, “Application-aware aggregation and traffic engineering in a converged packet-circuit network,” in *Proc. Opt. Fib. Comm. Conf. and Exposition and the National Fib. Optic Eng. Conf.*, Los Angeles, pp. 1-3, 2011.

[4] M. Rifai *et al.*, “MINNIE: An SDN world with few compressed forwarding rules,” *Comput. Net.*, vol. 121, pp 185-207, July 2017.

[5] S. Luo, H. Yu, and L. M. Li, “Fast incremental flow table aggregation in SDN,” in *Proc. Int. Conf. Comput. Commun. Networks, ICCCN*, Shanghai, pp. 1-8, 2014.

[6] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, “Infinite CacheFlow in Software-Defined Networks,” in *Proc. HotSDN Workshop*, Chicago, pp. 175-180, 2014.

[7] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, “CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks,” in *Proc. Symp. on SDN Research*, Santa Clara, pp. 1-12, 2016.

[8] X. Li and W. Xie, “CRAFT : A Cache Reduction Architecture for Flow Tables in Software-Defined Networks,” in *Proc. IEEE Symp. on Computers and Comm. (ISCC)*, Heraklion, pp. 1-6, 2017.

[9] Z. Guo *et al.*, “JumpFlow: Reducing flow table usage in software-defined networks,” *Comput. Net.*, vol. 92, Part 2, pp. 300-315, 2015.

[10] J. F. Huang, G. Y. Chang, C. F. Wang, and C. H. Lin, “Heterogeneous Flow Table Distribution in Software-Defined Networks,” *IEEE Trans. Emerg. Top. Comput.*, vol. 4, no. 2, pp. 252 -261, July 2016.

[11] A. Marsico, R. Doriguzzi-Corin, and D. Siracusa, “Overcoming the memory limits of network devices in SDN-enabled data centers,” *in Proc. Symp. Integr. Netw. Serv. Manag.*, Lisbon, pp. 897-898, 2017.

[12] H. Li, S. Guo, C. Wu, and J. Li, “FDRC: Flow-driven rule caching optimization in software defined networking,” in Proc. *IEEE Int. Conf. Commun.*, London, pp. 5777-5782, 2015.

[13] “Open vSwitch.” [Online]. Available: http://openvswitch.org/.

[14] O. N. Foundation, “OpenFlow Switch Specification,” pp. 1-177, 2015.

[15] “ONOS.” [Online]. Available: http://onosproject.org/.

[16] "Data Plane Development Kit," [Online]. Available: http://dpdk.org/

Paper H: Analysis of Traffic Engineering capabilities for SDN-based Data Center Networks

A. Pilimon, A. Mimidis-Kentis, S. Ruepp and L. Dittmann, "Analysis of Traffic Engineering capabilities for SDN-based Data Center Networks", presented at *The Second IEEE international workshop on Software Defined Networks and Network Function Virtualization (SDN-NFV)*, Barcelona, Spain, 23-26 April, 2018, 6 p.

DOI: -

Analysis of Traffic Engineering capabilities for SDN-based Data Center Networks

Artur Pilimon, Angelos Mimidis Kentis, Sarah Ruepp and Lars Dittmann
Department of Photonics Engineering
Technical University of Denmark
{artpil, agmimi, srru, ladit}@fotonik.dtu.dk

***Abstract*— In the recent years more and more existing services have moved from local execution environments into the cloud. In addition, new cloud-based services are emerging, which are characterized by very stringent delay requirements. This trend puts a stress in the existing monolithic architecture of Data Center Networks (DCN), thus creating the need to evolve them. Traffic Engineering (TE) has long been the way of attacking this problem, but as with DCN, needs to evolve by encompassing new technologies and paradigms. This paper provides a comprehensive analysis of current DCN operational and TE techniques focusing on their limitations. Then, it highlights the benefits of incorporating the Software Defined Networking (SDN) paradigm to address these limitations. Furthermore, it illustrates two methodologies and addresses the scalability aspect of DCN-oriented TE, network and service testing, by presenting a hybrid physical-simulated SDN enabled testbed for TE studies.**

***Keywords*—Data Center; SDN; Traffic Engineering.**

I. INTRODUCTION

Over the years, Data Centers (DC) and their supporting network infrastructure have gone through several important stages of architectural transformation. The scope of this transformation was to create a more economically feasible, reliable and energy-efficient communication environment, capable of accommodating ever growing data storage, processing and distribution demands.

Data Center Networks (DCN) can undoubtedly be regarded as the digital backbone of the global economy. DCN provide the business and mission critical communication for a vast range of entities (e.g. governments, business enterprises and private users). This diversity of the userbase translates into fundamentally different communication requirements in terms of reliability and service availability and Quality of Service (QoS) guarantees (bandwidth, latency, on-demand resource scaling, speed of failure recovery, etc.). As a result, Data Center businesses (large DC operators, wholesale DC solution providers and enterprise-level operators of DC facilities) are continuously challenged to look for new and more efficient ways of operating the existing DCNs, as well as defining novel strategies of building new DC facilities.

In this paper, we provide an analysis of traffic engineering (TE) capabilities for Software Defined Networking (SDN) based DCN environments and highlight the advantages and shortcomings of different strategies. The purpose of this study is to provide a condensed analysis of the most important DCN-oriented TE approaches, reported in a variety of other extensive surveys [1][2][3][4][5][6], but which cover a very broad range of SDN TE areas. In addition, we illustrate two methodologies for evaluating the benefits of traffic engineering capabilities in SDN controlled DCNs. The remainder of this paper is organized as follows: Section II explores and analyzes the TE capabilities offered by SDN in the context of DCNs. This includes overview of the common DC operation strategies, existing TE methodologies and their limitations, as well as discussion of the benefits of applying principles of SDN in TE. Section III discusses the need for TE in modern DCNs, and section IV highlights two methodologies for testbed evaluation of SDN TE features. Section V concludes the paper.

II. DATACENTER NETWORK OPERATION STRATEGIES

The initiatives for finding efficient ways of operating the existing DCNs, as well as defining novel approaches of building new DC facilities, have developed into three main strategies, widely adopted by the DC industry, namely:

1) DC resource overprovisioning;
2) DC resource usage optimization;
3) Building business-specific custom DC solutions.

The main features of each of these approaches are briefly highlighted as follows. The first strategy (**DC resource overprovisioning**) tackles the DC resource demand problem, should this be compute, storage or network infrastructure, by dimensioning the DCN for the "worst-case" scenario. In this case, the maximum possible all-to-all communication and service provisioning (including processing and storage) demands are taken as an input to the network dimensioning "formula". As a result, excessive redundancy of the allocated resources is supposed to handle the communication and service provisioning demands; however, there is a range of factors, which play a definitive role in making this strategy unfeasible, both from the economic and technical point of view. First, the potential Capital Expenditures (CAPEX) may be too high. Second, network performance degradation can occur even in an overprovisioned DCN environment. For example, traffic congestion can occur due to inefficient scheduling of the traffic flows or memory buffer limitations of the network devices. Therefore, some network links may be heavily loaded (leading to highly variable delays, jitter, packet loss) whilst other may be underutilized, creating imbalance of resource usage.

The second approach (**DC resource usage optimization**) is utilizing different optimization techniques, originating from research and industrial best practices, with the overall goal of increasing the efficiency of DCN's resource utilization, improving the QoS for services and applications in a shared DCN, as well as reducing the operational costs (OPEX). In this context, Traffic Engineering (TE) is an integral part of the

network optimization philosophy. As reported in [8], the average resource utilization of DC resources was ranging from 17.76% - 60% for computing resources, up to 77.93% for internal memory resources and up to 75.28% for disk storage. However, in most cases, the DCN resources are highly underutilized. By applying suitable TE techniques we can eliminate or alleviate the resource usage inefficiency and performance degradation threats, in addition to a potential reduction of OPEX. Adoption of the Software Defined Networking (SDN) paradigm [9] in the context of TE in DCNs is offering great improvements compared to the currently used TE solutions, and introduces new opportunities to fully exploit the potentials of modern DCN infrastructures.

The third strategy (**building business-specific custom DC solutions**) was derived by the web-scale service providers, such as Google, Amazon, Facebook, Netflix and alike. These service providers developed custom DCN designs with tailored TE solutions for their operational business needs, rather than trying to adapt industry standard solution and deal with arising scalability, performance and efficiency problems. Such solutions include, for example, Google's Jupiter architecture [10] or Facebook's Fabric [11] or custom routing platforms. However, these solutions cannot be directly reused in any other DCN environment, because they are built for specific use-cases, dictated by business demands.

III. The Need For Traffic Engineering in Modern Data Center Networks

According to the Global Cloud Index forecast [12], global Data Center IP traffic (intra-DC, DC-to-DC and DC-to-user) is expected to increase 3-fold by 2020, reaching 15.3 zettabytes. Other trends, which have significant effect on the concentration and distribution of the global DC workloads, as well as growth of intra-DC traffic volumes are [12]:

1) Growth of the hyper-scale DCs (estimated 53% of total DC traffic);
2) Virtualization and Cloud computing growth (up to 92% of workloads will be processed in Cloud DCs by 2020);
3) Rise of public and private Clouds;
4) Fast proliferation of new types of applications and services with diverse resource requirements (Big Data Analytics, Internet of Things, Multimedia content, Map-Reduce processing models).

To overcome the challenges imposed by these factors, flexible and highly efficient network resource management and TE approaches are required. In the following sub-sections, we discuss the key objectives of TE in DCNs (*A*) and evolution of TE approaches (*B*), limitations of the current TE approaches and why they cannot be effectively used in the SDN-based DCNs (*C*), as well as emphasize the main benefits and potentials of SDN-based TE to optimize the resource usage in DCNs (*D*) and outline some open research challenges in the context of SDN (*E*).

A. The objectives of TE techniques for DCNs

TE can be seen as an iterative process of performance optimization, carried out as a combination of monitoring and measurement techniques, static or dynamic adjustment of core relevant operational parameters. The ultimate goal of this process is to reach and sustain a carefully defined performance objective. It is important to distinguish the main performance objectives, usually pursued by TE (optimization mechanisms); however, integration of several TE objectives into one mechanism can be unfeasible, because certain TE goals can be mutually exclusive (e.g., network performance in terms of latency/throughput and energy efficiency) [1] [4][6]:

Minimization of network congestion: This is one of the most important performance objectives in communication networks, DCNs in particular. Congestion is one of the most significant problems, which directly affects other associated performance metrics, such as packet loss, latency and jitter. The problem affects the operational state of the DCNs more than that of a Wide Area Network (WAN), due to higher concentration of traffic with highly variable, bursty profiles and sub-second lifetime of vast majority of the flows within the DC (East-West traffic). This means that the critical performance tweaking decisions must be made in a very fast and dynamic on-line fashion. This performance metric can be optimized by utilizing the multipath redundancy of DCN architectures and spreading the traffic over the DCN infrastructure, relocating the resources for established flows (e.g., by disaggregation of large flows) or performing resource scheduling and access control (e.g., granting access only to uncongested resources of the network).

Minimization of the end-to-end (E2E) delay: This is a critical parameter in the context of DCNs, since a dominant volume of traffic flows within a DCN are short-lived and small-sized ("mice flows"); as a result, a typical flow duration in a DCN may be a few orders of magnitude shorter than in a long-haul transport network. On the other hand, DCs may be hosting business critical applications and services with very stringent delay requirements. Thus, it is of uttermost importance to keep the upper bound of this metric as low as possible, e.g., by utilizing efficient flow scheduling/load balancing or routing algorithms, such as Constrained-Shortest Path First (CSPF), which uses E2E delay as a path selection constraint. This parameter is also correlated with another metric, widely used as a complementary QoS assessment parameter in multimedia transmission services, namely Quality of Experience (QoE).

Maximization of the Energy Usage Efficiency: The focus of this objective is on minimizing the energy consumption of the DCN infrastructure by applying a set of techniques, including flexible resource consolidation and workload migration to be able to power down the network (nodes, ports, line cards) and processing (servers, storage disks and arrays) elements, which have become idle. Other approaches may include application of advanced optical switching techniques within the DCN interconnect (intra-DCN) [13].

Optimization of the resource utilization: Bandwidth, packet buffer space as well as CPU (Central Processing Unit) processing capacity are critical resources, which must be used in the most efficient way. Unavailability of any of these DCN resources will directly impact the performance of the hosted applications and services, affected by the packet loss, increase in queueing delays and flow completion times. This can be achieved by, e.g., applying flexible flow scheduling mechanisms, such as Weighted constrained ECMP (Equal Cost Multi-Path) for weight-based flow distribution over

multiple available paths of the same cost, or by scheduling well characterized data transfers (e.g., backup flows, updates) at certain time of the day, when the availability of processing and transmission resources may be higher.

Minimization of the packet loss: This objective is a derivative of the global Congestion minimization objective, and an applied TE strategy largely depends on the detected root cause of the packet loss, since packets can be lost as a result of network congestion, protocol/algorithmic failures or queue management mechanisms (e.g., Drop-Tail or Active Queue Management) activated in the network devices.

B. The evolution of TE techniques and applicability in DCNs

Traditionally, considering data packet networks, there have been different TE approaches introduced, as described in RFC 3272 [14]. However, the communication networks were rapidly evolving and these techniques were not satisfying the new performance optimization requirements anymore, as highlighted by Awduche in [15].

One of the first successfully adapted TE strategies for the Internet traffic was a concept of overlay model/network, with the best example of this being IP over ATM (Asynchronous Transfer Mode) [16]. This was realized by the means of using virtual circuits and virtual path concepts, which allowed defining multiple virtual topologies over the physical infrastructure. However, the downside was the increased operational complexity of the networks, in addition to the fact that the circuits had to be pre-provisioned – limiting the flexibility of possible TE manipulations, especially for real-time applications and in the context of DCN scale factor.

Another important step in the evolution of TE was the introduction of the Shortest Path First (SPF) algorithms, which could take the Type of Service (ToS) [17] specifications into account when choosing routing paths. However, the limitation of this method led to unfair sharing of the network resources, where the SPF-based shortest paths were ending up being overloaded (congested), whilst other paths were severely underutilized. Next solution enabled better utilization of the multi-path redundancy in the infrastructures, with multiple sets of paths of equal cost (as can be found in DCN environments), called ECMP [18]. It allowed equal traffic splitting among multiple available shortest paths by using a flow hashing technique; this method is widely used (in different modifications) in DCN environments up to date.

Multi-Protocol Label Switching (MPLS) [7][19] emerged as a traffic forwarding architecture, with a goal of increasing the flexibility, performance and scalability of the network layer routing, where the core packet processing (classification, marking) functionality shifted to the edge of the MPLS domain, while performing fast packet switching based on label information. This introduced a whole range of more advanced TE capabilities, including the explicit routing, traffic disaggregation and aggregation by applying multi-tunneling (multiple levels of encapsulation).

Finally, a PCE-based (Path Computation Element) architecture was proposed by IETF for MPLS and Generalized MPLS (GMPLS) networks [20]. In this solution, there is a dedicated element (e.g. a server) responsible for path computation, supporting explicit routing (point-to-point and point-to-multipoint label switched paths, LSPs). Hence, it is being currently adapted and used for intra-domain, inter-domain and inter-layer TE applications, in Optically-switched networks in particular [21].

C. Limitations of the traditional state-of-the-art TE approaches in the context of DCNs

As mentioned in Section I, the operational performance requirements in DCNs are fundamentally different compared to the classical transport/WAN environments, and the shortcomings of the traditional TE approaches are described as follows (more extensive general overviews can be found in [1][4][6]):

Non-optimal path computation algorithms: Available research results in [22] show that even when using MPLS-TE solution with CSPF algorithms for path computation and network resource management, increased latency was observed in numerous experiments. This was a result of combined impact of the CSPF algorithms used, auto-bandwidth scaling functionality (part of MPLS-TE framework), which led to continuous path changes due to automatic bandwidth adjustments on the LSPs (depending on the variations in traffic demands). In DCNs, where there might be millions of simultaneous mostly short-lived connections, this strategy would introduce delayed TE decisions, which would continuously affect the network state and potentially result in suboptimal paths being chosen.

Unbalanced traffic disaggregation (splitting) ratios: There are two ways to perform traffic splitting, either per-packet or per-flow. In the former, multiple performance problems may occur due to the inherent properties of the TCP (Transmission Control Protocol, being the dominant transport protocol in DCNs), such as performance drop due to packet reordering and subsequent false fast retransmissions and transmission window reductions [23]. In addition, using ECMP for traffic splitting may lead to congestion on the shortest paths, since the traffic is equally spread based on flow hashing in a uniform fashion; hence, e.g., large and small flows may collide on the shortest chosen paths.

Slow update rates of the TE Databases (TED): In this situation the contents of the TED do not reflect the network state in real-time, which may be a critical requirement for a DCN environment with fast changes in flow dynamics (flow arrival and completion, burstiness, etc.). Even though the current PCE-based TE methods can address the limitations of the traditional MPLS-TE, it is still facing a real-time data consistency problem [20], because this element entirely relies on the information stored in the TED for path computation and optimization.

Long convergence time of distributed protocols [4]**:** The problem of all the discussed so far approached is that they use in-band signaling for resource management, consuming network resources. Also, both MPLS-TE and PCE-based solutions rely on the RSVP-TE protocol (resource ReSerVation Protocol), which is an extra delay component, because once the path is computed by PCE, this information needs to be propagated to all the nodes of the path, and the resources are reserved after this reservation is confirmed by all these nodes. This aspect poses a direct scalability problem of this TE mechanism and leads to variable convergence times of

the network state information, affecting the number of flows that can be accommodated in a DCN within a short time unit.

D. The benefits of SDN in addressing TE challenges in DCNs

SDN, as a relatively new networking paradigm, introduced a highly flexible framework, allowing for tackling of the TE challenges, discussed in sub-section *C*. The key functional mechanisms of SDN, which enable intelligent TE decisions to optimize operational performance of DCNs, are outlined as follows [1][4][6]:

Separation of the control and forwarding planes: This feature is a definitive factor, enabling the network programmability by offloading the control decisions from the network devices to a logically centralized control unit, called an SDN controller. Such functional separation eliminates extra resource consumption in the data plane, while the core control logic is enforced to the data plane via out-of-band control channel, using an OpenFlow [24] protocol or other supported Southbound D-CPI (Data-Control Plane Interface) protocol [4]. This factor enhances the scalability of the control plane, while facilitating fast data plane forwarding of traffic flows.

Logically centralized control plane: This means that the control plane is presented as a logically centralized abstraction, but different implementation strategies are possible, including the clustering of multiple SDN controllers together. Clustering of the control plane is a very important aspect in terms of scalability, since it allows for virtual slicing of underlying network resources, where each virtual network domain can be controlled independently or in a hierarchical fashion. The benefit of this is that the SDN control has a unified view of all the network resources, operational state of the components and traffic load in real-time. Hence, a single centralized entity can utilize all this information to perform path computation and flow rule placement based on the defined policies for a corresponding traffic type. In the context of DCNs, scalability of the control plane is a critical factor to handle multi-granular TE decisions at sub-second time scales. Hence, cluster-based deployment of multiple SDN controllers with efficient coordination and network state synchronization techniques is a reasonable approach for DCNs.

High network programmability: This feature is of utmost importance, since it allows dynamic flow rule installation in the network. In addition, SDN provides also much finer levels of granularity (multiple protocol headers and individual fields can be used) in the flow rule definition. This enables application of more advanced TE policies by, e.g., exploiting a multi-table architecture of the flow processing pipeline (logical flow processing sequence). It is possible to define a complex ruleset by utilizing three types of flow processing tables available, such as the standard Flow tables, group tables (flow group processing actions) and meter tables (for traffic shaping/rate control). SDN provides a flexible network abstraction model through a well-defined open source Southbound interface (SBI), allowing to build complex control applications via an extensible API (Application programming interface). When compared to closed and proprietary interfaces or vendor-specific legacy network switching and routing devices, SDN allows for more efficient exploitation of multi-path redundancy of DCNs and performance isolation in multi-tenant architectures of Cloud DCNs to account for individual QoS requirements of common Cloud service models (SaaS, PaaS, IaaS, etc.).

Advanced network monitoring and performance measurements: The SDN control framework can be used to collect (and calculate) fine-grained statistical performance data from the network devices by means of native SDN statistics polling or integration of traditional (e.g., SNMP, Simple Network Management Protocol) or more advanced (e.g., sFlow, NetFlow, or jFlow) network monitoring techniques. In this way it is possible to create a sophisticated network monitoring framework by combining the strengths of these solutions. In addition, the collected historical measurement data can be used as an input for use of ML/DL (Machine Learning/Deep Learning) techniques to define TE objectives based on the predicted performance evolution, so that timely localized TE adjustments can made instead of global dynamic TE actions, since the complexity, diversity and massive volume of DCN traffic profiles may hinder the practical feasibility of dynamic TE enforcement.

E. Open research questions in SDN in the DCN context

In general, there have been multiple initiatives proposed in the research community [1][4], targeting TE in SDN-based DCNs to address the issues of operational complexity in SDNs in path computation [25], improve routing [26][27] and resource utilization efficiency [28]. Most of these solutions have been tested in virtualized network environments and with different available open source SDN controllers (OpenDaylight, Floodlight, etc.). The main message in all works is that SDN offers more flexibility in testing of any developed TE methods, since the key logic is developed as a loadable module in a centralized framework. However, this flexibility comes at a price, and there are multiple challenges for the SDN-based TE as well. For example, the following major open questions have been identified by Akyildiz in [3]:

Scalability and availability: It has been noticed that under higher flow arrival rates at the flow table resources available in switches (Ternary Content Addressable Memory (TCAM), software tables), installation of flow rules was accompanied by increased latency and latency spikes. Thus, efficient flow management (load balancing) solutions are needed for the Control and Data planes.

Multiple flow tables: As noted, certain SDN-enabled switches have only a single flow table built around TCAM. This type of memory is space limited, and large volumes of flow may lead to huge rule sets to be installed, limiting large scale deployment capabilities. Hence, multi-table implementations will improve the situation.

Reliability: For large-scale SDN deployments, a cluster-based approach is needed with operational and backup controllers. However, there is no standardized protocol (e.g., like OpenFlow) that would provide any mechanism for coordination between the primary and backup controllers, therefore limiting the possibilities for fault tolerance of the control plane.

Other important issues are consistency of the topology updates (problems due to duplicate flow entries, stalled entries, etc.) and accuracy of traffic analysis (e.g., Big Data sampling challenges).

IV. Analysis of Traffic Engineering Properties in an SDN-enabled Testbed Environment

One of the challenges of testing limitations of TE capabilities in DCN is the lack of available large-scale testbeds. In this section, we illustrate two methodologies, based on experimental studies, which can be combined to enable realistic and scalable TE testing framework. In the first, we feature a testbed composed of HP Aruba DCN SDN switches, an SDN-enabled Polatis open-space optical circuit switch, and an external SDN controller. This testbed, was used for the final demonstration of EU FP7 COSIGN project [29]. In addition, we also present a hybrid (physical-simulated) testbed (for Proof-of-Concept please refer to [30]; a system without SDN capabilities is presented in [31]) for evaluating SDN TE characteristics at scale. We are using the hypercube structure illustrated in Fig. 1, due to its simplicity, flexibility and scalability properties.

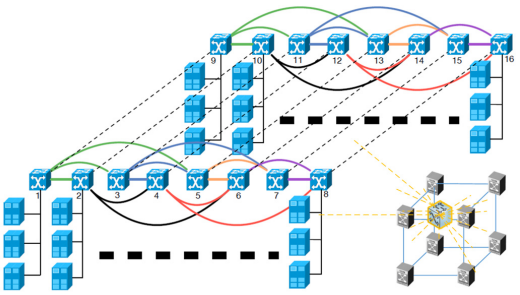


Fig. 1 Hypercube datacenter structure

The first study evaluates the latency figures, if SDN based TE is used for rerouting traffic via the Polatis switch to provide optical shortcuts in case of certain applications or network conditions. As a starting point multiple flows are routed via the shortest hypercube path. Once the threshold on the link reaches a specified or dynamically derived threshold (e.g. 70%), the traffic is re-routed through the optical shortcut. This TE feature is highly beneficial for low latency applications, load balancing and survivability in datacenters. The study steps are illustrated in Fig. 2.

The second test environment consists of a hybrid real-simulated setup, that allows linking of real networking equipment with a simulation model, in order to evaluate large scale SDN TE capabilities, that otherwise may not be possible due to CAPEX limitations [30]. The setup is shown in Fig. 3, with a model to real world connectivity exemplified in Fig. 4.

Implementation of a Hypercube-based simulation model consists of network switches, supporting Open Flow 1.3.0 protocol. The simulation model is connected to the real physical hardware (network equipment, servers) and an external SDN controller via specialized virtual System-in-the-Loop (SITL) gateways linked to the physical network interface(s) of a workstation/server, which the simulation environment is running on.

The applied simulation-based approach brings many benefits when evaluating the scalability of networks, datacenters and communication systems in general. The primary advantage is the real-time operation mode of the simulation model, meaning that realistic behavior and timing of processes within the simulated SDN-enabled network nodes are preserved. It is critical for these settings to be real-time to obtain accurate information of packet conversion times and buffering latencies at the boundary points between the real and simulated parts of the hybrid simulation model. The second advantage refers to the support of SDN principles via the implementation of the OpenFlow protocol in the simulated nodes.

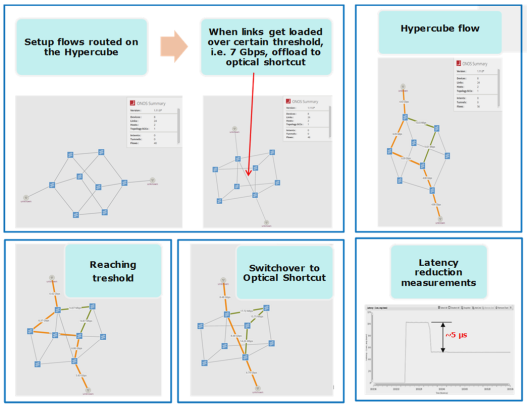


Fig. 2 Latency reduction study: SDN-based TE

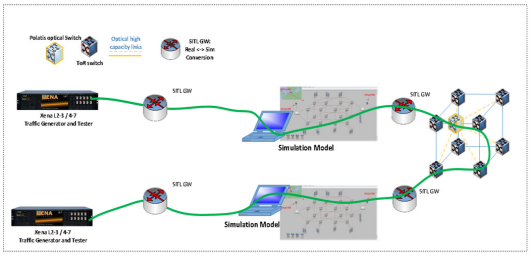


Fig. 3 Hybrid real-simulated SDN-enabled DCN testbed setup

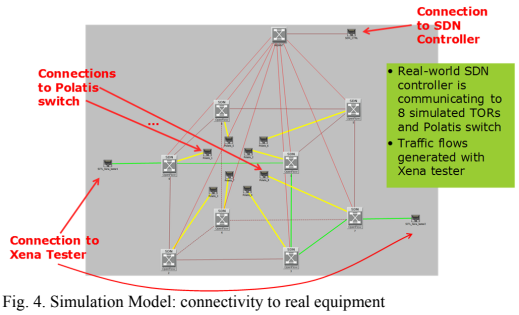


Fig. 4. Simulation Model: connectivity to real equipment

These features provide substantial flexibility in terms of reconfiguration, as the control logic only needs to be modified in the SDN controller instead of every individual simulated node, placement of additional real or hybrid components, as well as an increase of the scale of the network interconnect by

extending the simulation model. This approach is highlighted in Fig. 5, emphasizing that the scalability property can be achieved using advanced simulation and modeling techniques, which would allow us to assess SDN TE solutions in a repeatable manner with a greater degree of control over the infrastructure, configuration and traffic generation.

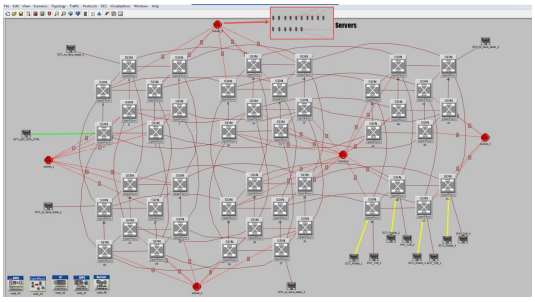


Fig. 5 A simulation model using an Incomplete Hypercube structure with 44 nodes and links to external DCN equipment [31]

V. Conclusions

In this work we have discussed the key limitations of the traditionally used TE approaches, taking into consideration the operational performance requirements in DCNs, due to completely different traffic profiles, with burstiness, critical bandwidth or ultra-low delay requirements, in addition to the sub-second lifetime of the dominant volume of all the traffic flows. These limitations include non-optimal path computation algorithms, slow TE Database convergence times, which is a critical requirement for the real-time control plane decisions. SDN-based TE solutions are discussed and the main benefits are distinguished with regards to how this new paradigm can greatly benefit the highly virtualized DCNs challenged by the new scalability, resource usage and operational efficiency requirements. The existing limitations of SDN-based approach are summarized as well.

We furthermore present two methodologies that can be used to evaluate traffic engineering capabilities in SDN. One is based on available hardware, whereas the other allows for large-scale datacenter evaluation by combining real equipment and a simulation environment.

References

[1] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in software defined networks," *Comput. Networks*, vol. 71, pp. 1–30, June 2014.

[2] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A Survey on Data Center Networking (DCN): Infrastructure and Operations," *IEEE Comm. Surv. and Tutor.*, vol. 19, no. 1. pp. 640–656, 2017.

[3] I. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, no. 3, pp. 52–58, June 2016.

[4] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, "A Survey on the Contributions of Software-Defined Networking to Traffic Engineering," *IEEE Comm.. Surv. Tutor.*, vol. 19, no. 2, pp. 918–953, May 2017.

[5] J. Zhang, F. Ren, and C. Lin, "Survey on transport control in data center networks," *IEEE Netw.*, vol. 27, no. 4, pp. 22–26, Aug. 2013.

[6] M. Noormohammadpour and C. S. Raghavendra, "Datacenter Traffic Control: Understanding Techniques and Trade-offs," *IEEE Comm.. Surv. Tutorials*, vol. PP, no. 99, pp. 1–34, Dec. 2017.

[7] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. Mcmanus, "Requirements for Traffic Engineering Over MPLS," RFC 2702 1999.

[8] X. Sun, N. Ansari, and R. Wang, "Optimizing Resource Utilization of a Data Center," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 4, pp. 2822–2846, Nov. 2016.

[9] P. A. Morreale, *Software Defined Networking: Design and Deployment*. CRC Press, 2014, pp. 1-186.

[10] A. Singh *et al.*, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," in *Proc. ACM SIGCOMM*, pp. 183–197, London, UK, 2015.

[11] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," in *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 123–137, 2015.

[12] Cisco, "Cisco Global Cloud Index : Forecast and Methodology, 2015–2020," *White Paper*. pp. 1–29, 2016.

[13] A. Pilimon, A. Zeimpeki, A. M. Fagertun, and S. Ruepp, "Energy Efficiency Benefits of Introducing Optical Switching in Data Center Networks," in *Proc. ICNC*, Santa Clara, CA, USA, 2017, pp. 891–895.

[14] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering," RFC 3272, 2002.

[15] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Commun. Mag.*, vol. 37, no. 12, pp. 42–47, Dec. 1999.

[16] R. Cole, D. Shur, and C. Villamizar, "IP over ATM: A Framework Document," RFC 1932, 1996.

[17] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, 1998.

[18] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992, 2000.

[19] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, 2001.

[20] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," RFC 4655, 2006.

[21] T. Tsuritani, M. Miyazawa, S. Kashihara, and T. Otani, "Optical path computation element interworking with network management system for transparent mesh networks," in Proc. *OFC/NFOEC*, San Diego, CA, USA, 2008, pp. 1–10.

[22] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. Maltz, "Latency inflation with MPLS-based traffic engineering," in *Proc. Acm Sigcomm Conference*, Berlin, Germany, 2011, pp. 463–472.

[23] K. C. Leung, V. O. K. Li, and D. Yang, "An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 522–535, Apr. 2007.

[24] Open Networking Foundation, "OpenFlow Switch Specification 1.4.0 (Wire Protocol 0x05)," 2013.

[25] L. A. Rocha, "Framework for Traffic Engineering of SDN Data Paths," *Adv. Appl. Sci.*, vol. 1, no. 2, pp. 37–45, Oct. 2016.

[26] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano, "Traffic Engineering with Segment Routing: SDN-Based Architectural Design and Open Source Implementation," *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN*, Bilbao, Spain, pp. 111–112, 2015.

[27] K. T. Dinh, S. Kukliński, W. Kujawa, and M. Ulaski, "MSDN-TE: Multipath Based Traffic Engineering for SDN," in *ACIIDS 2016. Lecture Notes in Computer Science*, 2016, vol. 9622, pp. 630–639.

[28] M. Khoshbakht, M. Tajiki, and B. Akbari, "SDTE : Software Defined Traffic Engineering for Improving Data Center Network Utilization," *Int. J. Inf. Commun. Technol. Res.*, vol. 8, no. 1, pp. 15–26, Mar. 2016.

[29] "COSIGN: Combining Optics and SDN In next Generation data centre Networks," Public Deliverable 5.5, 2017. [Online]. Available: http://www.fp7-cosign.eu

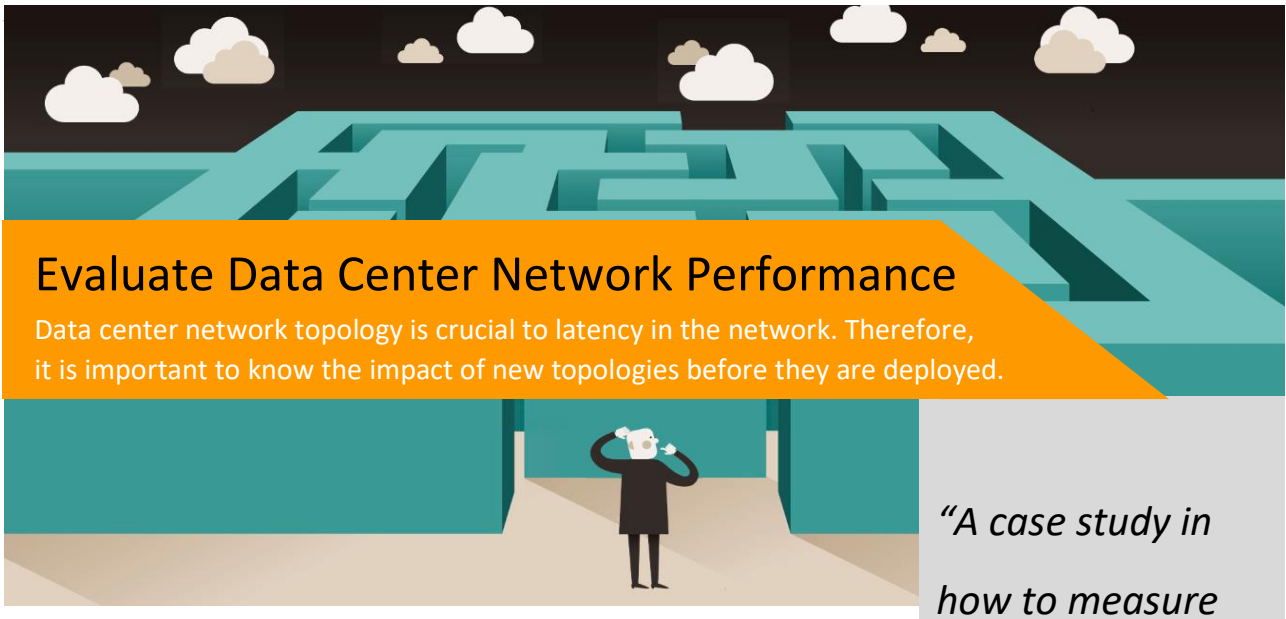
[30] S. Ruepp, A. Pilimon, J. Thrane, M. Galili, M. Berger, and L. Dittmann, "Combining Hardware and Simulation for Datacenter Scaling Studies," in Proc. *ONDM*, Budapest, Hungary, 2017, pp. 1–6.

[31] A. Pilimon and S. Ruepp, "A Hybrid Testbed for Performance Evaluation of Large-Scale Datacenter Networks," *accepted for presentation at ICNC 2018, Maui, HI, USA*.

Paper I: Evaluate Data Center Network Performance

O. Sørensen* (Xena Networks ApS), **A. Pilimon*** (DTU) and S. Ruepp (DTU), "Evaluate Data Center Network Performance", *Technical White Paper*, pp. 1-11, Jan. 2018.[Online]. Available: https://xenanetworks.com/evaluate-data-center-network-performance-white-paper/.

DOI: -



Evaluate Data Center Network Performance

Data center network topology is crucial to latency in the network. Therefore, it is important to know the impact of new topologies before they are deployed.

"A case study in how to measure the latency of data center network topologies with an SDN-based control framework and Xena test equipment"

OVERVIEW

Data centers are the foundation for numerous services that many people today take for granted. Use of these services grows exponentially, causing large organizations to continuously establish new, huge data centers to support the increasing demands.

Data centers contain numerous servers connected through a data center network, which is usually built with layer 2 switches and layer 3 routers. The topology of the data center network is crucial for latency in the data communication to and from the data center and between servers in the data center.

Tests can be conducted to measure latency and other performance parameters for different data center network topologies. It is however important that tests can be repeated and reproduced to have comparable information from the tests.

There are, of course, many topologies that can be used for data center networks. At DTU Fotonik, Department of Photonics Engineering, scientists evaluate data center network topologies with an SDN-based (Software-Defined Networking) control framework measuring network performance – primarily latency. This can be used to plan data center scaling by testing how a new topology will function before changes are made.

Data center network performance can, of course, be tested with Xena Networks solutions. To generate test signals with stateful TCP traffic the Xena Networks testers supporting layer 4-7 - XenaScale and XenaAppliance – are the obvious choice. Testing at lower layers is supported by the XenaBay and XenaCompact test chassis equipped with relevant test modules.

WHITE PAPER

EVALUATE DATA CENTER NETWORK PERFORMANCE

Contents

WHITE PAPER

INTRODUCTION

There are thousands of data centers world-wide and they are indispensable for the modern, web oriented society. Data centers are fundamental for numerous cloud based services, which many people today take for granted like streaming, social media networking, e-commerce, on-line banking, Anything-as-a-Service (XaaS) and many more. Large companies such as Google, Microsoft, Apple and Facebook run data centers to provide their offerings to customers and they continue to build new, hyperscale data centers to support exponentially growing demands.

Data centers are built with numerous servers providing a service that is available to end-users (or "clients"). The servers are connected via a data center network, enabling communication between the end users and the servers ("north-south" traffic). The data center network also enable communication between the servers inside the data center ("east-west" traffic).

The data center networks are built with layer 2 switches and layer 3 routers. Early networks could have a hierarchical tree-like topology as depicted in figure 1.

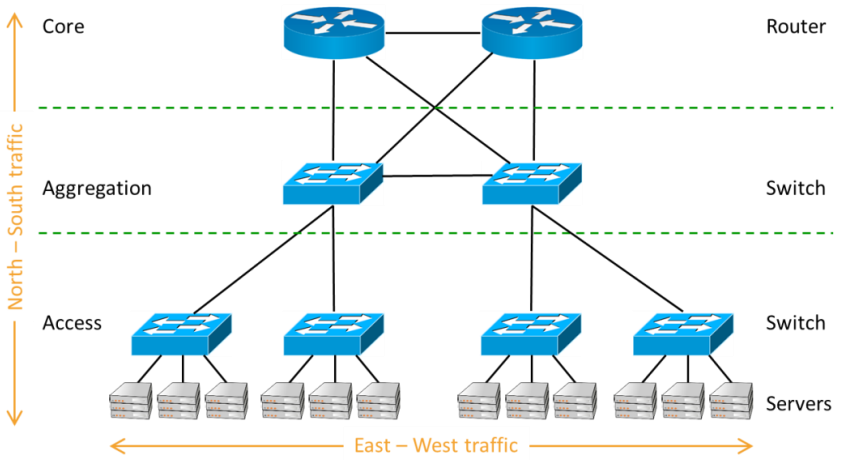


Figure 1: Early data center network topology

With the simple topology in figure 1 there is a risk of congestion in the aggregation layer as several access switches share an aggregation switch. In addition, some of the "east-west" traffic will experience increased latency when the server-to-server traffic must go through several aggregation layer switches. Such issues will increase if more layers are added to support more servers in the data center.

Over the years, the simple data center network topology in figure 1 has developed into other topologies, including the two-layer "leaf-spine" topology, having "leaf" switches forming an Access layer and "spine" switches in the aggregation layer as illustrated in figure 2. With the leaf-spine topology every leaf switch is directly connected to all spine switches in a mesh. Hereby the "east-west" traffic only needs to go through one spine switch, minimizing the latency. The leaf-

spine topology can include spine switches that only handle the "east-west" traffic, reducing the risk for congestion. The leaf-spine topology is useful for data centers with more "east-west" traffic than "north-south" traffic.

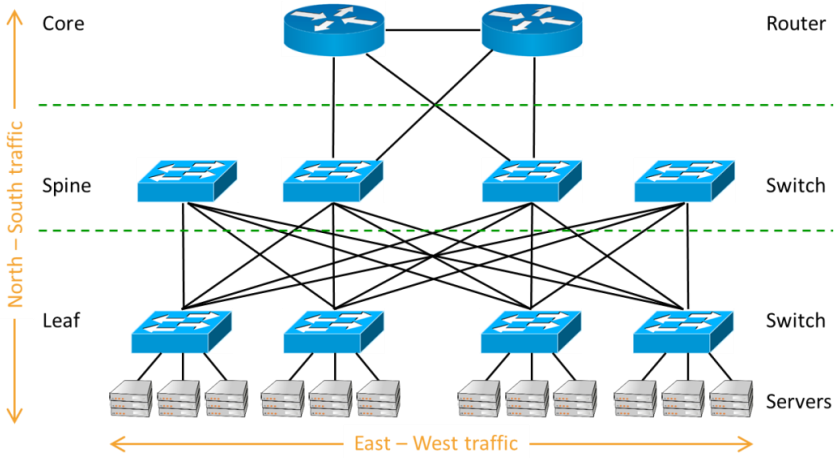


Figure 2: Leaf-spine data center network topology

In figure 1 and figure 2 layer 2 switches are used in the spine/aggregation layer. In some implementations layer 3 routers are used instead. Furthermore, virtualization is widely spread in data centers, meaning that while the logical structure of the data center may be as shown in figures 1 and 2, the actual hardware components may look different.

Software-Defined Networking (SDN) can also be used in data centers. SDN enables very flexible and agile configurations, changing the behavior of network elements by updating their flow tables, which control the traffic forwarding. Hereby the traffic flow through the data center network can dynamically and efficiently be adapted to changing requirements.

Data centers are not isolated entities. In addition to communicating with end-users they can also communicate together through optical Data Center Interconnect (DCI) links over the distance in between them. The increasing need for capacity on these links is driving the development of high speed optical systems.

Latency is an important parameter when designing data center networks. Some applications can be extremely latency sensitive e.g. stock market trading and banking transactions. Therefore, to support latency sensitive applications, data centers network topologies should minimize latency.

Tests can be executed to measure performance and, in particular, latency for different data center network topologies. It is, however, important that tests can be repeated and reproduced in order to have comparable information from the tests.

There are of course more topologies that can be used for data center networks. At DTU Fotonik, Department of Photonics Engineering scientists evaluate data center network topologies with an SDN-based (Software Defined Networking) control framework and measure the performance – primarily latency. This can be used to plan data center scaling by testing the impact of a new topology before changes are made in the data center network.

DATA CENTER NETWORK TESTBEDS: PERFORMANCE EVALUATION AND TESTING

Data centers (DC) and their supporting network infrastructure have become a backbone of the global digital economy, which needs to provide reliable and scalable communication services, while continuously being challenged in terms of the energy efficiency and resource utilization, Quality of Service (QoS) and performance isolation, architectural scalability and cost effectiveness. Hence, it is of paramount importance to conduct timely and comprehensive testing and performance evaluation of the existing and new technologies and protocols, network architectures and traffic engineering (TE) approaches. In addition to analytical modelling and simulations, this can be achieved by applying diverse and innovative research methodologies for testing of real data center network equipment in a realistic communication context (e.g., generating DC-specific traffic profiles, conducting experiments on large-scale DC network testbeds) so that the obtained results could be applicable at scale.

Building a large-scale data center just for experimental research purposes may not be a feasible option, both from the footprint and financial point of view. However, assembling a smaller scale, but sufficiently functional data center testbed, consisting of a subset of real data center network equipment (e.g., electrical and optical switches) with powerful SDN-based control framework and high-performance traffic generators with useful stress-testing capabilities, is a more realistic and flexible approach.

DTU Fotonik – Department of Photonics Engineering at the Technical University of Denmark – is actively using the capabilities of the Layer 2-3 (Xena Bay) and Layer 4-7 (Xena Scale) network testers for their data center research. One of the recent studies carried out was focusing on the experimental evaluation of a direct-connection topology, namely a Hypercube structure, applied as a data center network interconnect, enhanced with optical bypass switching capabilities. High level network connectivity diagrams of two configuration scenarios of a data center testbed are presented in figure 3 (8-node Cube) and figure 4 (16-Hypercube), respectively. All the data center network switches, both optical and electrical, are configured and controlled via an external SDN controller. Nevertheless, one of the most challenging tasks faced in this research activity was to create a functional data center-oriented traffic generation framework to be able to carry out different conformance and performance tests. After multiple different approaches have been tried out, the solution was found by combining the functional capabilities of both Layer 2-3 and Layer 4-7 testers. The reasons are outlined as follows:

1. We were looking for a solution, which could help us achieve two main goals: a) to be able to perform high-speed stress-testing of particular data center network segments and devices by loading these components with large number of traffic flows with sustainable data rates, and

b) to be able to create customized traffic profiles with traffic flow groups of different duration, data volume size as well as configurable network, transport and application layer properties. The former objective was achieved by using Layer 2-3 Xena Bay platform, which also provides great means of collecting accurate per-stream performance statistics (e.g., average, maximum, minimum latency and jitter, packet loss, etc.). The latter requirement was satisfied by using Layer 4-7 Xena Scale tester, which allowed us to mix different groups of stateful (TCP connections) and stateless (UDP flows) traffic flows and configure relevant parameters, such as TCP window sizes, congestion control, segment or flow sizes.

2. When conducting network testing at scale, test automation capabilities are becoming critically important, because this results in significant time savings to define and configure various test scenarios as well as process and analyze the gathered results. We used the available CLI-based scripting interface and developed Python scripts to control the tests, gather statistics and visualize the results. Thus, by just changing a set of command-line arguments, a completely different set of tests can be configured automatically. This aspect greatly extends the possibilities of test repeatability and reproducibility of the results.

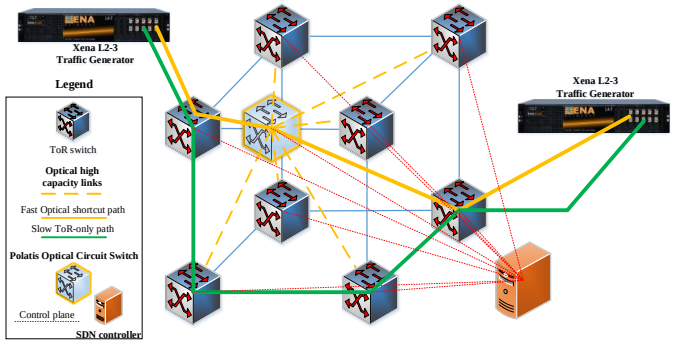


Figure 3: DCN Testbed setup. 8-node Cube with Optical bypass and Xena Bay L2-3 tester

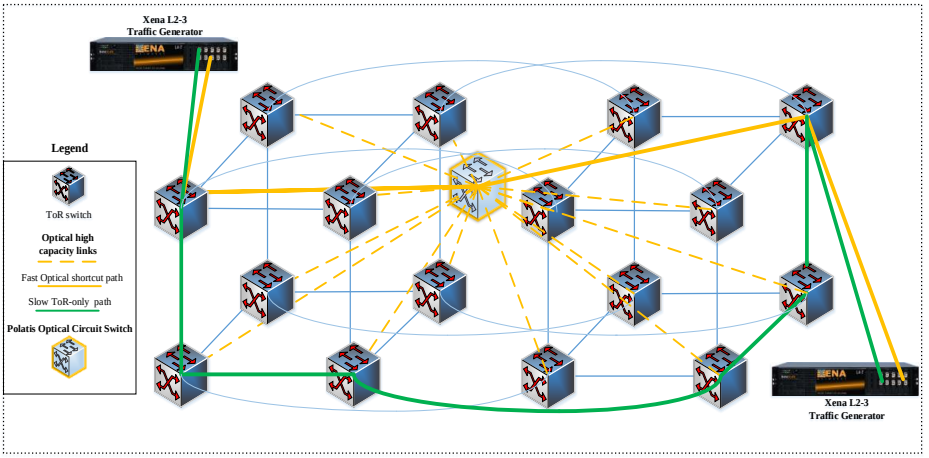


Figure 4: DCN Testbed setup. 16-Hypercube with Optical bypass and Xena Bay L2-3 tester

WHITE PAPER

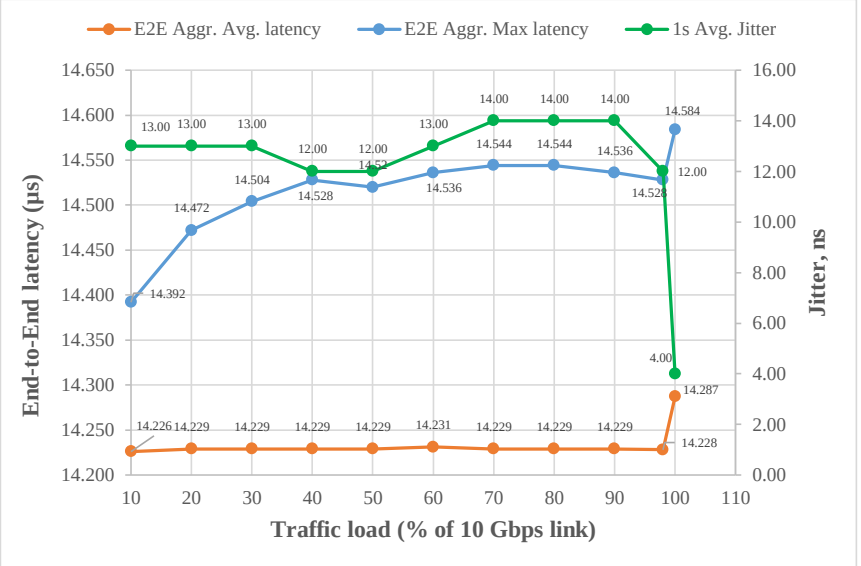


Figure 5: End-to-End latency measurement test results obtained with Xena Bay L2-3 tester. Topology: 8-node Cube, Polatis OCS. Legend: E2E – End-to-End, Aggr. – Aggregate, Avg. – Average, s – second, Max – Maximum

3. Another important aspect in favor of using an advanced hardware traffic generator and tester, instead of a software traffic generator, is the resolution (granularity) and accuracy of the measurement results. Considering latency measurements in the context of data center networking, this becomes particularly problematic when using software generators, since the crafted packets are reaching the Network Interface Card (NIC) through the shared kernel space of the underlying operating system (OS), and time-expensive interrupt-based processing is greatly limiting the maximum achievable packet generation rate. Software-level timestamping accuracy is another issue, since simple software generators are limited by the clock granularity of the underlying OS, while hardware counterparts, such as Xena testers, offer more accurate hardware timestamping capabilities and allow tracking statistics at different level of detail, e.g., as it can be seen in figure 5.

DATA CENTER BENCHMARKING

Following the massive deployment of data centers, the Internet Engineering Task Force (IETF) in August 2017 published a couple of documents on data center benchmarking:

- RFC 8238 Data Center Benchmarking Terminology
- RFC 8239 Data Center Benchmarking Methodology

WHITE PAPER

As the titles indicate, RFC 8238 presents new terminology in relation to benchmarking of data center network equipment, while RFC 8239 defines how to perform the benchmarking tests of switches and routers that are used in a data center.

Data center traffic dynamically changes over time. The traffic will in periods predominantly be “north-south” between the servers and a client outside the data center and in other periods be more ”east-west” oriented between servers in the data center. Traffic can be a mix of TCP and UDP flows, and can be a result of point-to-multipoint or multipoint-to-multipoint communication patterns, commonly found in data centers. Traffic may be sensitive to latency or throughput and all kinds of traffic can exist simultaneously in a data center network element. Previously IETF has published several documents on network element and network benchmarking, including:

- RFC 2544 Benchmarking Methodology for Network Interconnect Devices
- RFC 2889 Benchmarking Methodology for LAN Switching Devices
- RFC 3918 Methodology for IP Multicast Benchmarking

RFC 8239 have test cases based on the above RFCs. In addition, it includes test cases that better than the above RFCs represent the wide range of traffic conditions that can exist in a data center. RFC 8239 test cases include:

Test	Description
Line-Rate Testing	A "maximum rate" tests for the performance values for throughput, packet drop, latency and jitter. Tests are conducted as a port-pair test and as a full-mesh test
Buffering Testing	Measuring the DUT buffer size under various traffic conditions
Microburst Testing	Identify the maximum amount of packet bursts that a DUT can sustain under various configurations
Head-of-Line Blocking (HOLB)	Examine a DUT’s behavior in case of HOLB and measure packet loss caused by HOLB, which occurs when packets are held up by the first packet ahead waiting to be transmitted to a different output port
Incast Stateful and Stateless Traffic	Measure TCP Goodput (retransmissions excluded) and latency under various traffic conditions. The test simulates a mix of stateful (TCP) flows requiring high goodput and stateless (UDP) flows requiring low latency

Table 1: RFC 8239 tests

The RFC 8239 tests require that several (in some cases all) ports of the DUT are connected to a traffic generator, as illustrated in figure 6.

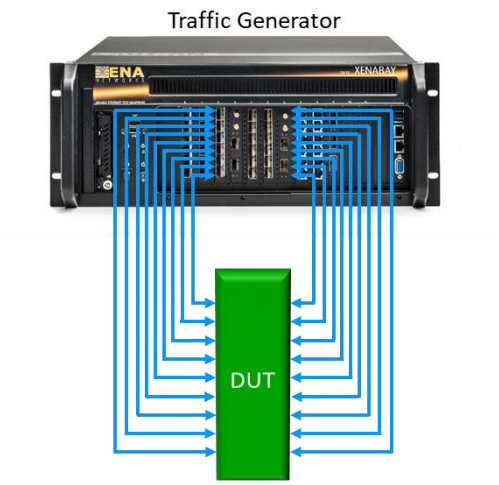


Figure 6: Typical RFC 8239 test setup

DTU FOTONIK (DEPARTMENT OF PHOTONICS ENGINEERING)

At DTU Fotonik, researchers work with multiple aspects of light (photons), in every layer were light can be used and controlled. Approximately 220 researchers work at DTU Fotonik, including around 90 PhD students. Taught programs are B.Sc. Network Technology and IT, M.Sc. in Telecommunications and M.Sc. in Photonics Engineering. In the Network Technologies and Service Platforms group at DTU Fotonik, the research is focused on four main directions, such as data centers (including SDN), fronthaul/backhaul solutions for mobile networks, IoT and core networks. In particular, the group has been leading the recently completed EU FP7 project COSIGN: Combining Optics and SDN In next Generation datacenter Networks (2014 – 2017). The group has sophisticated lab facilities to carry out a variety of datacenter network experiments.

XENA NETWORKS DATA CENTER TEST SOLUTIONS

Data Center Network Performance can of course be tested with Xena Networks test solutions. To generate test signals with stateful TCP traffic the Xena Networks testers supporting layer 4-7 - XenaScale and XenaAppliance – are the obvious choice. Testing at lower layers is supported by the XenaBay and XenaCompact test chassis equipped with relevant test modules.

TESTING ABOVE LAYER 3

Figure 7: The powerful Xena Networks Layer 4-7 testers XenaScale and XenaAppliance

WHITE PAPER

Xena Network's XenaScale and XenaAppliance can be used to generate TCP, HTTP/TCP and UDP traffic streams simultaneously. In addition, both products offer stateful end-to-end testing of network appliances such as switches, firewalls, routers, NAT routers, proxies, load-balancers, bandwidth shapers and more. The platform is also suitable to characterize entire network infrastructure performance for TCP. Top features include:

- Wire-speed stateful TCP traffic generation and analysis with extreme performance
- TLS performance testing with different cipher suites and certificates
- Application emulation with real-world application traffic mixes enabled by XenaAppMix
- Replay captured traffic at scale
- Configuration and tuning of Ethernet, IP and TCP header fields for advanced traffic scenarios
- Stateful TCP connection
- HTTP get/put/head/post
- Extensive live stats and test reports
- 1G – 10G Ethernet interfaces
- 40G Ethernet interfaces (XenaScale)
- High port density – up to 12 x 10 GigE (XenaScale)
- Configurable allocation of processing resources to Ethernet test ports
- Wire-speed traffic capture
- Switched and routed network topologies, NAT support
- Export packet capture to industry standard pcap/Wireshark

TESTING UP TO LAYER 3

Figure 8: The versatile and powerful Xena Networks Layer 2-3 testers XenaBay and XenaCompact

Testing at lower layers is supported by the XenaBay and XenaCompact test chassis equipped with relevant test modules, which can support data rates up to 100 Gbps. Up to 12 test modules can be installed in the XenaBay chassis. Based on Xena's advanced architecture, XenaBay and XenaCompact equipped with relevant test modules are proven solutions for Ethernet testing at layers 2 and 3. Advanced test scenarios can be performed using the free Xena test applications:

WHITE PAPER

- **XenaManager-2G** test software is used to configure and generate streams of Ethernet traffic between Xena test equipment and Devices Under Test (DUTs) and analyze the results
- **Xena2544** offers full support for the 4 test types specified in RFC 2544: Throughput, Latency, Frame loss and Back-to-back frames; Jitter (Frame Delay Variation) is also supported
- **Xena1564** provides full support for both the configuration and performance test types described in Y.1564 for complete validation of Ethernet Service Level Agreements (SLAs) in a single test
- **Xena2889** is an application for benchmarking the performance of Layer 2 LAN switches in accordance with RFC 2889
- **Xena3918** makes it easy to create, edit and execute all test types specified in RFC 3918. RFC 3918 describes tests for measuring and reporting the throughput, forwarding, latency and IGMP group membership characteristics of devices that support IP multicast protocols

TEST AUTOMATION

The Xena Networks L4-7 and L2-3 test solutions have a scripting Command Line Interface (CLI), which is ideal for test automation. The user can create a script, which defines a test sequence that can be repeated as often as required, providing reproduceable results.

CONCLUSION

Data centers contain numerous servers providing a service for end-users. The servers are connected via a data center network, which is built with layer 2 switches and layer 3 routers. The topology of the data center network is crucial for latency in the communication between servers and end users and in server-to-server communication.

Many topologies can be used for data center networks. At DTU Fotonik, Department of Photonics Engineering scientists evaluate the performance of different data center network topologies with an SDN-based (Software Defined Networking) control framework, measuring primarily latency. This can be used to plan data center scaling by testing how a new topology will function before changes are made.

DTU Fotonik
Department of Photonics Engineering

Data Center Network performance can of course be tested with Xena Networks test solutions. To generate test signals with stateful TCP traffic the Xena Networks testers supporting layer 4-7 - XenaScale and XenaAppliance – are the obvious choice. Testing at lower layers is supported by the XenaBay and XenaCompact test chassis equipped with relevant test modules.

WHITE PAPER

Bibliography

[1] W. Stallings. *Data And Computer Communications*. Eighth Edition. Upper Saddle River, NJ: Pearson Prentice Hall, 2007, pp. 655–693.

[2] *Akamai's [state of the internet] Q1 2017 report*. Tech. rep. 1. Akamai, 2017, p. 64.

[3] *Internet Growth Statistics*. Internet World Stats: usage and population statistics. Dec. 2017. URL: https://www.internetworldstats.com/emarketing.htm;%20https://www.internetworldstats.com/stats.htm.

[4] *Cisco Visual Networking Index (VNI): The Zettabyte Era—Trends and Analysis*. June 2017. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[5] Jakob Nielsen. *Nielsen's Law of Internet Bandwidth*. Nielsen Norman Group (NN/g). 1998 - 2018. URL: https://www.nngroup.com/articles/law-of-bandwidth/.

[6] *Keeping Up With the Ever Increasing Demand for Internet Bandwidth*. OTelco. Dec. 2016. URL: https://www.otelco.com/keeping-pace-internet-bandwidth-demand/.

[7] *Keeping Pace with Nielsen's Law*. CableLabs. Sept. 2016. URL: https://www.cablelabs.com/how-will-law-treat-injuries-caused-autonomous-vehicles.

[8] *50 Years of Moore's Law: Fueling Innovation We Love and Depend On*. Intel Corp. 2015. URL: https://www.intel.com/content/www/us/en/silicon-innovations/moores-law-technology.html.

[9] Vladimir Oksman, Rainer Strobel, Xiang Wang, Dong Wei, Rami Verbin, Richard Goodson, and Massimo Sorbara. "The ITU-T's new G.fast standard brings DSL into the gigabit era". In: *IEEE Communications Magazine* 54.3 (2016), pp. 118–126. ISSN: 01636804. DOI: 10.1109/MCOM.2016.7432157.

[10] *IEEE Std 802.16-2017 (Revision of IEEE Std 802.16-2012) - IEEE Standard for Air Interface for Broadband Wireless Access Systems*. IEEE SA. 2017. URL: https://standards.ieee.org/findstds/standard/802.16-2017.html.

[11] *LTE-Advanced*. 3GPP. 2013. URL: http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced.

[12] *5G: Networks of the future*. Ericsson. 2018. URL: https://www.ericsson.com/en/networks/trending/hot-topics/5g-networks.

[13] *Wireless Broadband Alliance: driving next wireless experience*. Wireless Broadband Alliance. 2018. URL: https://www.wballiance.com/.

[14] *Wi-Fi Alliance® publishes 2018 Wi-Fi® predictions*. Wi-Fi Alliance. 2018. URL: https://www.wi-fi.org/.

[15] Bret Swanson. *The Exponential Internet*. US Chamber of Commerce. 2013. URL: https://www.uschamberfoundation.org/bhq/exponential-internet.

[16] *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*. Tech. rep. Cisco, 2017, pp. 1–17.

[17] Cisco. *Cisco Global Cloud Index : Forecast and Methodology , 2016 - 2021*. Tech. rep. 2018, pp. 1–41.

[18] *The Case for the Metro Core*. Nokia. Jan. 2014. URL: https://insight.nokia.com/case-metro-core.

[19] *Infrastructure update: evolution of the Dropbox backbone network*. Dropbox Inc. Sept. 2017. URL: https://blogs.dropbox.com/tech/2017/09/infrastructure-update-evolution-of-the-dropbox-backbone-network/.

[20] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113. ISSN: 00010782. DOI: 10.1145/1327452.1327492. arXiv: 10.1.1.163.5292. URL: http://portal.acm.org/citation.cfm?doid=1327452.1327492.

[21] *The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology*. Tech. rep. Gaithersburg, MD, 2011.

[22] *Software-Defined Networking (SDN) Definition*. Open Networking Foundation. 2011. URL: https://www.opennetworking.org/sdn-definition/.

[23] *Where the world meets DevOps*. MediaOps LLC. 2018. URL: https://devops.com/about/.

[24] *A Look at Key SDN Deployments*. SDxCentral, LLC. June 2016. URL: https://www.sdxcentral.com/articles/analysis/key-sdn-deployments/2016/06/.

[25] David Geer. *Five reasons IT pros are not ready for SDN investment*. TechTarget, Geer Communications. 2015. URL: https://searchsdn.techtarget.com/feature/Five-reasons-IT-pros-are-not-ready-for-SDN-investment.

[26] *Bandwidth Management: Internet Society Technology Roundtable Series | Internet Society*. Tech. rep. Internet Society, 2012, pp. 1–13. URL: http://www.internetsociety.org/doc/bandwidth-management-internet-society-technology-roundtable-series.

[27] OFCOM. *Traffic Management and 'net neutrality'*. Tech. rep. September. OFCOM, 2010, pp. 1–68. URL: http://stakeholders.ofcom.org.uk/binaries/consultations/net-neutrality/summary/netneutrality.pdf.

[28] *Net Neutrality challenges*. Oct. 2015. URL: https://ec.europa.eu/digital-agenda/en/net-neutrality-challenges%7B%5C#%7DArticle.

[29] *Net neutrality: How we got from there to here - CNET*. Feb. 2015. URL: http://www.cnet.com/news/net-neutrality-from-there-to-here/.

[30] D Clark, S Bauer, and William Lehr. "Measurement and Analysis of Internet Interconnection and Congestion". In: *The Research Conference on Communication, Information and Internet Policy*. 2014, pp. 1–16. URL: http://www.caida.org/publications/papers/2014/measurement%7B%5C_%7Danalysis%7B%5C_%7Dinternet%7B%5C_%7Dinterconnection/measurement%7B%5C_%7Danalysis%7B%5C_%7Dinternet%7B%5C_%7Dinterconnection.pdf.

[31] *Network Congestion Definition*. The Linux Foundation Project. Nov. 2005. URL: http://www.linfo.org/congestion.html.

[32] *Congestion*. Techopedia Inc. 2018. URL: https://www.techopedia.com/definition/18506/congestion-networks.

[33] Scott Hogg. *10GE and Network Oversubscription Ratios*. IDG Communications Inc. Dec. 2009. URL: https://www.networkworld.com/article/2232813/cisco-subnet/cisco-subnet-10ge-and-network-oversubscription-ratios.html.

[34] "Campus QoS Design". In: *Cisco AVVID Network Infrastructure Enterprise Quality of Service Design Solutions Reference Network Design*. Second. Cisco Press, 2005. Chap. 2, pp. 1–32.

[35] J Postel. *Transmission Control Protocol, RFC 793*. Tech. rep. September. 1981, pp. 1–86. URL: http://tools.ietf.org/html/rfc793.

[36] J. Postel. *User Datagram Protocol, RFC 768*. Tech. rep. IETF, 1980, pp. 1–3. DOI: 10.1016/B978-0-12-374541-5.50016-X.

[37] *Netflix takes up 9.5 percent of upstream traffic on the North American Internet*. ARSTechnika. Nov. 2014. URL: https://arstechnica.com/information-technology/2014/11/netflix-takes-up-9-5-of-upstream-traffic-on-the-north-american-internet/.

[38] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. "Inside the Social Network's (Datacenter) Network". In: *SIGCOMM Comput. Commun. Rev.* Vol. 45. 4. New York, NY, USA: ACM, Aug. 2015, pp. 123–137. DOI: 10.1145/2829988.2787472. URL: http://doi.acm.org/10.1145/2829988.2787472.

[39] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Holzle, Stephen Stuart, and Amin Vahdat. "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network". In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), pp. 183–197. ISSN: 0146-4833. DOI: 10.1145/2785956.2787508. URL: http://doi.acm.org/10.1145/2785956.2787508.

[40] Shaojun Zou, Jiawei Huang, Yutao Zhou, Jianxin Wang, and Tian He. "Flow-Aware Adaptive Pacing to Mitigate TCP Incast in Data Center Networks". In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), pp. 2119–2124. DOI: 10.1109/ICDCS.2017.170. URL: http://ieeexplore.ieee.org/document/7980158/.

[41] Yang Qin, Weihong Yang, Yibin Ye, and Yao Shi. "Analysis for TCP in data center networks: Outcast and Incast". In: *Journal of Network and Computer Applications* 68 (2016), pp. 140–150. ISSN: 10958592. DOI: 10.1016/j.jnca.2016.04.014. URL: http://dx.doi.org/10.1016/j.jnca.2016.04.014.

[42] *Fixed Mobile Convergence for the Enterprise*. Cisco. Oct. 2007. URL: https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-fixed-mobile-convergence-solutions/net_implementation_white_paper0900aecd806ec474.html.

[43] *Fixed mobile convergence*. Nokia Networks. 2018. URL: https://networks.nokia.com/intelligent-access/fixed-mobile-convergence.

[44] Partha Kanuparthy and Constantine Dovrolis. "ShaperProbe: end-to-end detection of ISP traffic shaping using active methods". In: *Proceedings of the 2011 ACM SIGCOMM ...* 2011, pp. 473–482.

[45] *ISP Interconnection and its Impact on Consumer Internet Performance*. Tech. rep. 2014.

[46] Hadi Asghari, Michel Eeten, and Milton Mueller. *Unravelling the Economic and Political Drivers of Deep Packet Inspection*. 2012.

[47] S. Bradner and J. McQuaid. *Benchmarking Methodology for Network Interconnect Devices, RFC 2544*. Tech. rep. IETF, 1999, pp. 1–31. arXiv: arXiv:1011.1669v3.

[48] L. Avramov and J. Rapp. *Data Center Benchmarking Methodology, RFC 8239*. Tech. rep. IETF, 2017, pp. 1–19.

[49] *Testing of Next Generation Networks*. ITU-T, 2011, pp. 1–52.

[50] Matheus A. Cavalcante, Helder A. Pereira, and Raul C. Almeida. "SimEON: an open-source elastic optical network simulator for academic and industrial purposes". In: *Photonic Network Communications* 34.2 (2017), pp. 193–201.

[51] *EstiNet Network Simulator and Emulator*. EstiNet Technologies Inc. 2011-2018. URL: http://www.estinet.com/ns/?page_id=21140.

[52] Michael Tighe. *DCSim: A Data Centre Simulation Tool for Evaluating Dynamic Virtualized Resource Management*. 2012. URL: https://github.com/digs-uwo/dcsim.

[53] *CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. Cloud Computing, Distributed Systems (CLOUDS) Laboratory, School of Computing, and Information Systems, The University of Melbourne. 2009-2018. URL: http://cloudbus.org/cloudsim/.

[54] *OMNeT plus plus: Discrete Event Simulator*. OpemSim Ltd. 2001-2018. URL: https://www.omnetpp.org/.

[55] *The Network Simulation - NS2*. URL: http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[56] *NS-3*. NS3 Consortium. 2009-2018. URL: https://www.nsnam.org/.

[57] Zhangxi Tan, Zhenghao Qian, Xi Chen, Krste Asanovic, and David Patterson. *DIABLO : Simulating Datacenter Network at Scale using FPGAs Datacenter Network Architecture Overview*. Berkeley, CA, USA, 2013. URL: www.hpts.ws/papers/2013/DIABLO.pdf.

[58] *PlanetLab: an Open platform for developing, deploying, and accessing planetary-scale services*. The PlanetLab Consortium. 2003-2018. URL: https://www.planet-lab.org/.

[59] *OnaLab: Future Internet Testbed*. 2018. URL: https://onelab.eu/.

[60] *GEANT Testbeds Service: Integrated virtual network environments to support advanced research*. GEANT. 2018. URL: https://www.geant.org/Services/Connectivity_and_network/Pages/GEANT_Testbeds_Service.aspx.

[61] *M-Lab: Open Internet Measurement*. Measurement Lab Consortium. 2018. URL: https://www.measurementlab.net/about/.

[62] Cristian Hernandez Benet, Robayet Nasim, Kyoomars Alizadeh Noghani, and Andreas Kassler. "OpenStackEmu - A Cloud Testbed Combining Network Emulation with OpenStack and SDN". In: *The 14th Annual IEEE Consumer Communications and Networking Conference*. January. 2017, pp. 566–568.

[63] *GENI (Global Environment for Network Innovations): exploring networks of the future*. supported by NSF. 2018. URL: http://www.geni.net/.

[64] *FEDERICA: Federated E-Infrastructure Dedicating to European Researchers Innovating in Computer Networks Architectures*. GARR Consortium. 2009. URL: http://www.fp7-federica.eu/about/related.php.

[65] Villy B. Iversen. *Teletraffic Engineering and Network Planning*. Kgs. Lynbgy, Denmark, 2010.

[66] Bernard Gendron. “Decomposition methods for network design”. In: *Procedia - Social and Behavioral Sciences* 20 (2011), pp. 31–37.

[67] *Ostinato: Network Traffic Generator and Analyzer*. 2017. URL: https://ostinato.org/.

[68] Salvatore Sanfilippo. *Hping - Active Network Security Tool*. 2006-2018. URL: http://www.hping.org/.

[69] *Scale Factor: Is Your Data Center Cloud-Ready?* Spirent Communications Inc. 2015. URL: https://www.spirent.com/Blogs/Networks/2015/September/How_to_test_your_cloud_with_Spirent_HyperScale.

[70] *L2-3 (Xena Bay) and L4-7 (Xena Scale) Test Platforms*. URL: http://www.xenanetworks.com.

[71] *Spirent Technology Solutions*. Spirent Communications Inc. 2018. URL: https://www.spirent.com/Products.

[72] *Test Architecture*. Ixia. 2018. URL: https://www.ixiacom.com/solutions/test-architecture.

[73] *Amazon Virtual Private Cloud*. Amazon Web Services. 2018. URL: https://aws.amazon.com/vpc/.

[74] Enrico Masala, Antonio Servetti, Simone Basso, and Juan Carlos De Martin. “Challenges and issues on collecting and analyzing large volumes of network data measurements”. In: *Advances in Intelligent Systems and Computing* 241 (2014), pp. 203–212. ISSN: 21945357. DOI: 10.1007/978-3-319-01863-8_23.

[75] *Processing: What to record?* European Council for Nuclear Research (CERN). 2018. URL: https://home.cern/about/computing/processing-what-record.

[76] Kaushik Kumar Ram, Ian C. Fedeli, Alan L. Cox, and Scott Rixner. “Explaining the impact of network transport protocols on SIP proxy performance”. In: *ISPASS 2008 - IEEE Int. Symp. Perform. Anal. Syst. Softw.* 2008, pp. 75–84. ISBN: 9781424422326. DOI: 10.1109/ISPASS.2008.4510740.

[77] Junxian Huang, Feng Quian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. “An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance”. In: *ACM SIGCOMM Conf.* 2013, pp. 363–374. ISBN: 9781450320566. DOI: 10.1145/2534169.2486006.

[78] Arash Molavi Kakhki. “Rigorous Evaluation of Performance and Policy Impacts of Transport Protocols and In-Network Devices”. PhD Thesis. Northeastern University Boston, The College of Computer and Information Science, Sept. 2017, pp. 1–130.

[79] R. Braden. *Requirements for Internet Hosts - Communication Layers, RFC 1122*. Tech. rep. October. 1989, pp. 1–116.

[80] P Hurtig, W John, and Anna Brunstrom. “Recent Trends in TCP Packet-Level Characteristics”. In: 2011, pp. 49–56. ISBN: 9781612081335.

[81] Craig Labovitz, Danny Mcpherson, Scott Iekel-Johnson, and Mike Hollyman. *Internet Traffic Trends: A View from 67 ISPs*. 2009.

[82] Darren Anstee. *Trends in Internet Traffic Patterns*. 2010.

[83] Dong Jin Lee, Brian E Carpenter, and Nevil Brownlee. “Media Streaming Observations: Trends in UDP to TCP Ratio”. In: *International Journal on Advances in Systems and Measurements* 3.3 (2010), pp. 147–162.

[84] L-A. Larzon, M. Degermark, S. Pink, Ed. L-E. Jonsson, and Ed. G. Fairhurst. *The Lightweight User Datagram Protocol (UDP-Lite), RFC 3828*. Tech. rep. IETF, 2004, pp. 1–12.

[85] Adam Langley, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Alistair Riddoch, Wan-Teh Chang, Zhongyi Shi, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, and Ian Swett. “The QUIC Transport Protocol: Design and Internet-Scale Deployment”. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM ’17*. 2017, pp. 183–196. ISBN: 9781450346535. DOI: 10.1145/3098822.3098842. URL: http://dl.acm.org/citation.cfm?doid=3098822.3098842.

[86] J Iyengar and I Swett. *QUIC Loss Detection and Congestion Control, draft-ietf-quic-recovery-08*. Tech. rep. IETF, 2017, pp. 1–13.

[87] R Hamilton, J Iyengar, I Swett, and A Wilk. *QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2 draft-tsvwg-quic-protocol-01*. Tech. rep. 2015.

[88] J Postel. *Internet Protocol (IP)*. 1981.

[89] M Allman, V Paxson, and E Blanton. *TCP Congestion Control, RFC 5681*. Tech. rep. Internet Engineering Task Force (IETF), 2009, pp. 1–18. DOI: 10.17487/rfc5681. URL: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf%7B%5C%%7D5Cnhttp://www.hjp.at/doc/rfc/rfc5681.html%20https://www.rfc-editor.org/info/rfc5681.

[90] D. Borman, B Braden, and V. Jacobson. *TCP Extensions for High Performance, RFC 7323*. Tech. rep. IETF, 2014, pp. 1–49.

[91] T Henderson, S Floyd, A. Gurtov, and Y. Nishida. *The NewReno Modification to TCP’s Fast Recovery Algorithm, RFC 6582*. 2012.

[92] M. Allman. *TCP Congestion Control with Appropriate Byte Counting (ABC), RFC 3465*. RFC. IETF, 2003.

[93] M. Allman, H. Balakrishnan, and S. Floyd. *Enhancing TCP's Loss Recovery Using Limited Transmit, RFC 3042*. Tech. rep. Internet Engineering Task Force (IETF), 2001, pp. 1–9. DOI: 10.17487/rfc3042. URL: https://www.rfc-editor.org/info/rfc3042.

[94] S Floyd, A. Arcia, D Ros, and J. Iyengar. *Adding Acknowledgement Congestion Control to TCP Abstract, RFC 5690*. Tech. rep. IETF, 2010, pp. 1–33.

[95] IEEE. *IEEE 802.3 ETHERNET WORKING GROUP*. 2018. URL: http://www.ieee802.org/3/ (visited on 02/13/2018).

[96] T. Socolofsky and C. Kale. *A TCP/IP Tutorial, RFC 1180*. Tech. rep. IETF, 1991, pp. 1–28.

[97] Hossein Bidgoli. *The Internet Encyclopedia*. 1st. Wiley, 2004. ISBN: 0471222011.

[98] J. Postel. *DOD STANDARD TRANSMISSION CONTROL PROTOCOL*. RFC 761. IETF, 1980.

[99] W Stevens. *TCP/IP Illustrated, Volume 1: Protocols*. Addison-Wiley, 1994.

[100] V. Jacobson and Michael J. Karels. "Congestion avoidance and control". In: *ACM SIGCOMM Computer Communication Review* 18.4 (1988), pp. 314–329. ISSN: 01464833. DOI: 10.1145/52325.52356. arXiv: arXiv:1011.1669v3. URL: http://portal.acm.org/citation.cfm?doid=52325.52356.

[101] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis. *Increasing TCP's Initial Window, RFC 3390*. Tech. rep. Internet Engineering Task Force (IETF), 2002, pp. 1–15. DOI: 10.17487/rfc6928. arXiv: arXiv:1011.1669v3. URL: https://www.rfc-editor.org/info/rfc6928.

[102] J. Chu, N Dukkipati, Y. Cheng, and M Mathis. *Increasing TCP's Initial Window, RFC 6928 (Experimental)*. Tech. rep. IETF, 2013, pp. 1–24. DOI: 10.17487/rfc6928. arXiv: arXiv:1011.1669v3. URL: https://www.rfc-editor.org/info/rfc6928.

[103] M Allman, V Paxson, and W Stevens. *TCP Congestion Control*. 1999.

[104] Artur Pilimon. "Analysis of Multiple High Speed TCP Connections". MA thesis. Technical University of Denmark, 2014, pp. 1–203.

[105] S Floyd, T Henderson, and A Gurtov. *The NewReno Modification to TCP's Fast Recovery Algorithm*. 2004.

[106] M. Mathis, J Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgment Options, RFC 2018*. Tech. rep. IETF, 1996, pp. 1–12. DOI: 10.17487/rfc2018. arXiv: arXiv:1011.1669v3. URL: https://www.rfc-editor.org/info/rfc2018.

[107] Kevin Fall and Sally Floyd. "Simulation-based comparisons of Tahoe, Reno and SACK TCP". In: *ACM SIGCOMM Computer Communication Review* 26.3 (1996), pp. 5–21. ISSN: 01464833. DOI: 10.1145/235160.235162. URL: http://portal.acm.org/citation.cfm?doid=235160.235162.

[108] S Floyd, M Handley, and J Padhye. “A Comparison of Equation Based and Aimd Congestion Control”. In: (2000). URL: http://www.aciri.org/tfrc/aimd.pdf.

[109] E. Altman, K. Avrachenkov, and C. Barakat. “TCP network calculus: the case of large delay-bandwidth product”. In: *Proceedings.Twenty-First Annu. Jt. Conf. IEEE Comput. Commun. Soc.* 2002, pp. 417–426. ISBN: 0-7803-7476-2. DOI: 10.1109/INFCOM.2002.1019284. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1019284.

[110] J Mo, R J La, V Anantharam, and J Warland. “Analysis and Comparison of TCP Reno and Vegas”. In: In Proceedings of INFOCOM. 1999.

[111] S. Floyd. *HighSpeed TCP for Large Congestion Windows, RFC 3649 (Experimental)*. Tech. rep. IETF, 2003, pp. 1–34.

[112] Yee-Ting Ting Li, Douglas Leith, and Robert N. Shorten. “Experimental Evaluation of TCP Protocols for High-Speed Networks”. In: *IEEE/ACM Transactions on Networking* 15.5 (2007), pp. 1109–1122. ISSN: 10636692. DOI: 10.1109/TNET.2007.896240.

[113] K. Ramakrishnan, S Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168*. Tech. rep. Internet Engineering Task Force (IETF), 2001, pp. 1–64. DOI: 10.17487/rfc3168. arXiv: arXiv:1011.1669v3. URL: https://www.rfc-editor.org/info/rfc3168.

[114] E. Blanton, M. Allman, K. Fall, and L Wang. *A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP, RFC 3517*. Tech. rep. IETF, 2003, pp. 1–13.

[115] E. Blanton, M. Allman, L. Wang, I. Jarvinen, M. Kojo, and Y. Nishida. *A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP, RFC 6675*. Tech. rep. IETF, 2012, pp. 1–15. DOI: 10.17487/rfc6675. URL: https://www.rfc-editor.org/info/rfc6675.

[116] N. Spring, D Wetherall, and D Ely. *Robust Explicit Congestion Notification (ECN) Signaling with Nonces, RFC 3540*. Tech. rep. IETF, 2003, pp. 1–13.

[117] S Floyd. *Limited Slow-Start for TCP with Large Congestion Windows, RFC 3742*. Tech. rep. IETF, 2004, pp. 1–7.

[118] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig. *Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP), RFC 5827*. Tech. rep. Internet Engineering Task Force (IETF), 2010, pp. 1–15. DOI: 10.17487/rfc5827. URL: https://www.rfc-editor.org/info/rfc5827.

[119] Nandita Dukkipati, Matt Mathis, Yuchung Cheng, and Monia Ghobadi. *Proportional rate reduction for TCP, RFC 6937 (Experimental)*. Tech. rep. Internet Engineering Task Force (IETF), 2013, pp. 1–16. DOI: 10.1145/2068816.2068832. URL: http://dl.acm.org/citation.cfm?doid=2068816.2068832.

[120] S Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. *An Extension to the Selective Acknowledgement (SACK) Option for TCP, RFC 2883*. Tech. rep. IETF, 2000, pp. 1–17. DOI: 10.17487/rfc2883. arXiv: arXiv:1011.1669v3. URL: https://www.rfc-editor.org/info/rfc2883.

[121] P. Sarolahti, M. Kojo, K. Yamamoto, and M. Hata. *Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP, RFC 5682*. Tech. rep. IETF, 2009, pp. 1–19.

[122] Ed. D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe. *Open Research Issues in Internet Congestion Control, RFC 6077 (Informational)*. Tech. rep. IETF, 2011, pp. 1–51. DOI: 10.1007/s13398-014-0173-7.2. arXiv: 9809069v1 [arXiv:gr-qc]. URL: http://www.ncbi.nlm.nih.gov/pubmed/15003161%7B%5C%%7D5Cnhttp://cid.oxfordjournals.org/lookup/doi/10.1093/cid/cir991%7B%5C%%7D5Cnhttp://www.scielo.cl/pdf/udecada/v15n26/art06.pdf%7B%5C%%7D5Cnhttp://www.scopus.com/inward/record.url?eid=2-s2.0-84861150233%7B%5C&%7DpartnerID=tZOtx3y1.

[123] M Duke, R Braden, W. Eddy, E. Blanton, and A. Zimmermann. *A Roadmap for Transmission Control Protocol (TCP) Specification Documents, RFC 7414 (Informational)*. Tech. rep. IETF, 2015, pp. 1–57. DOI: 10.17487/rfc7414. arXiv: arXiv:1011.1669v3. URL: http://tools.ietf.org/html/rfc1323%7B%5C%%7D7D%7B%5C%%7D22%20https://www.rfc-editor.org/info/rfc7323.

[124] M. Welzl and D Ros. *A Survey of Lower-than-Best-Effort Transport Protocols, RFC 6297*. Tech. rep. IETF, 2011, pp. 1–18.

[125] M. Welzl and W Eddy. *Congestion Control in the RFC Series, RFC 5783 (Informational)*. Tech. rep. 2010, pp. 1–28.

[126] Nandita Dukkipati, Matt Mathis, Yuchung Cheng, and Monia Ghobadi. "Proportional rate reduction for TCP". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference - IMC '11*. Berlin, 2011, pp. 155–169. ISBN: 9781450310130. DOI: 10.1145/2068816.2068832. URL: http://dl.acm.org/citation.cfm?doid=2068816.2068832.

[127] Frank P. Kelly, A. K. Maulloo, and D. K.H. Tan. "Rate control for communication networks: Shadow prices, proportional fairness and stability". In: *J. Oper. Res. Soc.* 49.3 (1998), pp. 237–252. ISSN: 14769360. DOI: 10.1057/palgrave.jors.2600523.

[128] Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock. "Host-to-host congestion control for TCP". In: *IEEE Commun. Surv. Tutorials* 12.3 (2010), pp. 304–342. ISSN: 1553877X. DOI: 10.1109/SURV.2010.042710.00114.

[129] Don Smith. *Empirical Analysis of TCP Losses and Its Detection / Recovery Mechanisms*. Tech. rep. TR05-017. 2006, pp. 1–19.

[130] Kevin L Mills and James J Filliben. *Study of Proposed Internet Congestion Control Mechanisms*. Tech. rep. 500-282. NIST Special Publication 500-282. National Intsitute of Standards and Technology (NIST), 2010.

[131] L Xu, K Harfoush, and I Rhee. "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks". In: *IEEE INFOCOM, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE INFOCOM 2004. Hong Kong, 2004, pp. 1–11.

[132] Sangtae Ha and Injong Rhee. "CUBIC : A New TCP-Friendly High-Speed TCP Variant". In: *ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel*. Vol. 42. 5. Proceedings of the third PFLDNet Workshop. 2008, pp. 64–74. DOI: http://doi.acm.org/10.1145/1400097.1400105.

[133] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L Eggert, and R Scheffenegger. *CUBIC for Fast Long-Distance Networks, RFC 8312 (Informational)*. Tech. rep. 9. 2018, pp. 1–18.

[134] Tom Kelly. "Scalable TCP: improving performance in highspeed wide area networks". In: *SIGCOMM Comput. Commun. Rev.* 33.2 (2003), pp. 83–91.

[135] R. D.Leith, Shorten, and Hamilton Ins. "H-TCP protocol for high-speed long-distance networks". In: *PFLDnet*. Argonne, 2004, pp. 1–16. URL: http://www.hamilton.ie/net/htcp3.pdf.

[136] D. Leith. *H-TCP: TCP Congestion Control for High Bandwidth-Delay Product Paths*. 2008.

[137] Kun Tan, Jingmin Song, Qian Zhang, and Murari Sridharan. "Compound TCP : A Scalable and TCP-Friendly Congestion Control for High-speed Networks". In: *Proceedings 25th Conference on Computer Communications (InfoCom 2006)* (2006), pp. 23–29.

[138] D X Wei, C Jin, S H Low, and S Hegde. "FAST TCP: Motivation, Architecture, Algorithms, Performance". In: 14 No. 6 (2006), pp. 1246–1259.

[139] Cao Yuan, Liansheng Tan, Lachlan L H Andrew, Wei Zhang, and Moshe Zukerman. "A Generalized FAST TCP scheme". In: *Computer Communications* 31.14 (2008), pp. 3242–3249. ISSN: 01403664. DOI: 10.1016/j.comcom.2008.05.028.

[140] Mirja Kuhlewind. *TCP SIAD : Congestion Control supporting High Speed and Low Latency*. Tech. rep. December. 2016, pp. 1–15.

[141] Andrea Baiocchi, a P Castellani, and Francesco Vacirca. "YeAH-TCP: Yet another highspeed TCP". In: *5th PFLDnet Workshop*. 2007, pp. 37–42.

[142] Aleksandar Kuzmanovic, Edward W Knightly, and R Les Cottrell. "HSTCP-LP : A Protocol for Low-Priority Bulk Data Transfer in High-Speed High-RTT Networks". In: (2004), pp. 3–4.

[143] Jon Crowcroft and Philippe Oechslin. “Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP”. In: *ACM SIGCOMM Computer Communication Review* 28.3 (1998), pp. 53–69. ISSN: 01464833. DOI: 10.1145/293927.293930. arXiv: 9808004v1 [arXiv:cs]. URL: http://portal.acm.org/citation.cfm?doid=293927.293930.

[144] Masayoshi Nabeshima. “Performance evaluation of MulTCP in high-speed wide area networks”. In: *IEICE Transactions on Communications* E88-B.1 (2005), pp. 392–396. ISSN: 17451345. DOI: 10.1093/ietcom/E88-B.1.392.

[145] J Wang, J Wen, J Zhang, and Y Han. “TCP-FIT: An improved TCP congestion control algorithm and its performance”. In: *INFOCOM, 2011 Proceedings IEEE* (2011), pp. 2894–2902. ISSN: 0743-166X. DOI: 10.1109/INFCOM.2011.5935128.

[146] Jingyuan Wang, Jiangtao Wen, Jun Zhang, Zhang Xiong, and Yuxing Han. “TCP-FIT: An improved TCP algorithm for heterogeneous networks”. In: *Journal of Network and Computer Applications* 71 (2016), pp. 167–180. ISSN: 10958592. DOI: 10.1016/j.jnca.2016.03.020.

[147] C Casetti, M Gerla, S Mascolo, M Y Sanadidi, and R Wang. “TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks”. In: *Wireless Networks*. 2002, pp. 467–479.

[148] Dzmitry Kliazovich, Fabrizio Granelli, and Daniele Miorandi. “TCP Westwood+ enhancement in high-speed long-distance networks”. In: *IEEE International Conference on Communications*. Vol. 2. c. Istanbul, 2006, pp. 710–715.

[149] Shao Liu, Tamer Başar, and R. Srikant. “TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks”. In: *Performance Evaluation* 65.6-7 (2008), pp. 417–440. ISSN: 01665316.

[150] C Caini and R Firrincieli. “TCP Hybla: a TCP enhancement for heterogeneous networks”. In: *INTERNATIONAL JOURNAL OF SATELLITE COMMUNICATIONS AND NETWORKING* 22 (2004), pp. 547–566.

[151] N. Cardwell, Y Cheng, S. Hassas Yeganeh, and V Jacobson. *BBR Congestion Control, draft-cardwell-iccrg-bbr-congestion-control-00*. Tech. rep. 2018, pp. 1–34.

[152] Aleksandar Kuzmanovic and Edward W. Knightly. “TCP-LP: Low-priority service via end-point congestion control”. In: *IEEE/ACM Transactions on Networking* 14.4 (2006), pp. 739–752. ISSN: 10636692. DOI: 10.1109/TNET.2006.879702.

[153] S Floyd. *Metrics for the Evaluation of Congestion Control Mechanisms, RFC 5166*. Tech. rep. IETF, 2008, pp. 1–23. DOI: 10.17487/rfc5166. URL: https://www.rfc-editor.org/info/rfc5166.

[154] Mark Allman. “Comments on Bufferbloat”. In: *ACM Computer Communication Review* 43.March (2013), pp. 31–37. ISSN: 01464833. DOI: 10.1145/2427036.2427041.

[155] Sonia Belhareth, Dino Lopez-Pacheco, Lucile Sassatelli, Denis Collange, and Guillaume Urvoy-Keller. "Understanding synchronization in TCP cubic". In: *2014 26th International Teletraffic Congress, ITC 2014*. 2014. ISBN: 9780988304505. DOI: 10.1109/ITC.2014.6932941.

[156] H Rahman, K Giridhar, and G Raina. "Performance analysis of Compound TCP with AQM". In: *1th International Symposium and Workshops on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. 2013, pp. 492–499. ISBN: 9783901882548.

[157] Stefan Holmer, H Lundin, and Gaetano Carlucci. "A Google Congestion Control Algorithm for Real-Time Communication, draft-alvestrand-rmcat-congestion". 2015.

[158] S Floyd, M Handley, J. Padhye, and J. Widmer. *TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 5348*. Tech. rep. 2008, pp. 1–58.

[159] Dragana Damjanovic and Michael Welzl. "MulTFRC : Providing Weighted Fairness for Multimedia Applications (and others too !)" In: *ACM Computer Communication Review* 39.3 (2009), pp. 6–11. ISSN: 01464833.

[160] E. Kohler, M. Handley, and S. Floyd. *Datagram Congestion Control Protocol (DCCP), RFC 4340*. Tech. rep. IETF, 2006, pp. 1–129.

[161] A Ford, C. Raiciu, M. Handley, and O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses, RFC 6824*. Tech. rep. 2013, pp. 1–64. arXiv: arXiv:1011.1669v3.

[162] Y Pryadkin and D Katabi. "Specification for the Explicit Control Protocol (XCP), draft-falk-xcp-spec-03". 2007.

[163] Chia Hui Tai, Jiang Zhu, and Nandita Dukkipati. "Making large scale deployment of RCP practical for real networks". In: *Proceedings - IEEE INFOCOM* (2008), pp. 201–205. ISSN: 0743166X. DOI: 10.1109/INFOCOM.2007.285.

[164] Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, Shivkumar Kalyanaraman, Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, and Shivkumar Kalyanaraman. "One more bit is enough". In: *ACM SIGCOMM Computer Communication Review* 35.4 (2005), pp. 1–15. URL: http://portal.acm.org/citation.cfm?doid=1090191.1080098.

[165] Bartek Wydrowski, Moshe Zukerman, and Senior Member. "MaxNet : A Congestion Control Architecture". In: *IEEE Communications Letters* 6.11 (2002), pp. 512–514.

[166] Yueping Zhang, Derek Leonard, and Dmitri Loguinov. "Jetmax: Scalable max-min congestion control for high-speed heterogeneous networks". In: *Computer Networks* 52.6 (2008), pp. 1193–1219. ISSN: 13891286. DOI: 10.1016/j.comnet.2007.11.021.

[167] D. Leith, Rn Shorten, and G. McCullagh. “Experimental evaluation of Cubic-TCP”. In: *Proceedings of PFLDnet* (2007), pp. 1–9. ISSN: 0733-9445. DOI: http://dx.doi.org/10.1061/(ASCE)0733-9445(1996)122:3(228). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.9364%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[168] Ryo Oura and Saneyasu Yamaguchi. “Fairness Comparisons Among Modern TCP Implementations”. In: *26th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2012*. 2. 2012, pp. 909–914. ISBN: 9780769546520. DOI: 10.1109/WAINA.2012.167.

[169] Jacopo Chicco, Denis Collange, and Alberto Blanc. “Simulation Study of New TCP Variants”. In: *15th IEEE Symposium on Computers and Communications, ISCC*. The IEEE symposium on Computers and Communications. 2010, pp. 50–55.

[170] Sangtae Ha and Injong Rhee. “Taming the elephants: New TCP slow start”. In: *Computer Networks* 55.9 (2011), pp. 2092–2110.

[171] Sushant Rewaskar, Jasleen Kaur, and F. Donelson Smith. “A Performance Study of Loss Detection/Recovery in Real-world TCP Implementations”. In: *Proceedings - International Conference on Network Protocols, ICNP*. IEEE International Conference on Network Protocols (ICNP 2007). 2007, pp. 256–265. ISBN: 1424415888. DOI: 10.1109/ICNP.2007.4375856.

[172] S. Bhandarkar, A. L., M. Allman, and E Blanton. *Improving the Robustness of TCP to Non-Congestion Events, RFC 4653*. Tech. rep. 2006, pp. 1–18.

[173] Ka Cheong Leung, Victor O.K. Li, and Daiqin Yang. “An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges”. In: *IEEE Transactions on Parallel and Distributed Systems* 18.4 (2007), pp. 522–535. ISSN: 10459219. DOI: 10.1109/TPDS.2007.1011.

[174] Vishnu Konda and Jasleen Kaur. “RAPID: Shrinking the congestion-control timescale”. In: *Proceedings - IEEE INFOCOM* (2009), pp. 1–9. ISSN: 0743166X. DOI: 10.1109/INFCOM.2009.5061900.

[175] L S Brakmo and L L Peterson. “TCP Vegas: End to End Congestion Avoidance on a Global Internet”. In: *IEEE Journal on selected areas in Communications* 13.8 (Oct. 1995), pp. 1465–1480.

[176] S Floyd, M Allman, A. Jain, and P. Sarolahti. *Quick-Start for TCP and IP*. Tech. rep. IETF, 2007, pp. 1–82.

[177] J Mo, R La, V Anantharam, and J Walrand. “Analysis and Comparison of TCP Reno and TCP Vegas”. In: *Proc. IEEE INFOCOM*. 1999, pp. 1–19.

[178] Kenji Kurata, Go Hasegawa, and Masayuki Murata. “Fairness comparisons between TCP Reno and TCP Vegas for future deployment of TCP Vegas”. In: *INET*. 2000, pp. 1–20.

[179] Joel Sing and Ben Soh. “TCP new Vegas: Improving the performance of TCP Vegas over high latency links”. In: *Proceedings - Fourth IEEE International Symposium on Network Computing and Applications, NCA 2005*. IEEE, 2005, pp. 73–80. ISBN: 0769523269. DOI: 10.1109/NCA.2005.52.

[180] George W Dunlap, Samuel T King, Sukru Cinar, Murtaza A Basrai, and Peter M Chen. “TCP Nice: A Mechanism for Background Transfers”. In: *Symposium on Operating Systems Design and Implementation*. Boston, MA, USA, 2002, pp. 1–15.

[181] Yi Cheng Chan, Chia Liang Lin, Chia Tai Chan, and Cheng Yuan Ho. “CODE TCP: A competitive delay-based TCP”. In: *Computer Communications* 33.9 (2010), pp. 1013–1029. ISSN: 01403664. DOI: 10.1016/j.comcom.2010.01.007. URL: http://dx.doi.org/10.1016/j.comcom.2010.01.007.

[182] Andrea De Vendictis, Andrea Baiocchi, and Michela Bonacci. “Analysis and enhancement of TCP Vegas congestion control in a mixed TCP Vegas and TCP Reno network scenario”. In: *Performance Evaluation* 53.3-4 (2003), pp. 225–253. ISSN: 01665316. DOI: 10.1016/S0166-5316(03)00064-6.

[183] U Hengartner, J Bolliger, and T Gross. “TCP Vegas revisited”. In: *INFOCOM 2000*. Vol. 3. Zurich, Switzerland, 2000, pp. 1546–1555.

[184] T C P Performance Cubic and Ing Luis Marrone. “TCP Performance - CUBIC, Vegas & Reno”. In: *JCS&T* 13.1 (2013), pp. 1–8.

[185] Tuan Trinh, Balázs Sonkoly, and Sándor Molnár. “Revisiting FAST TCP Fairness”. In: *18th ITC Specialist Seminar on Quality of Experience*. 2008, pp. 1–8.

[186] Heying Zhang, Xunying Zhang, Baohua Fan, and Lisong Shao. “Adaptive FAST TCP”. In: *2010 Second International Conference on Future Networks*. 2010, pp. 114–118.

[187] Sushant Rewaskar, Jasleen Kaur, and Don Smith. *Why Don’t Delay-based Congestion Estimators Work in the Real-world?* Tech. rep. University of North Carolina at Chapel Hill, 2006.

[188] G McCullagh and DJ Leith. *Delay-based congestion control: Sampling and correlation issues revisited*. Tech. rep. Maynooth: University of Ireland, 2008, pp. 1–12. URL: http://www.hamilton.nuim.ie/net/correl%7B%5C_%7DToN.pdf.

[189] Ravi S. Prasad, Manish Jain, and Constantinos Dovrolis. “On the Effectiveness of Delay-Based Congestion Avoidance”. In: *Second International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*. 3. 2004, pp. 1–2.

[190] David Hayes. *Timing enhancements to the FreeBSD kernel to support delay and rate based TCP mechanisms*. Tech. rep. February. Melbourne: Swinburne University of Technology, 2010, pp. 1–4.

[191] A Gurtov and S Floyd. “Modeling wireless links for transport Protocols”. In: *Acm Sigcomm Computer Communication Review* 34.2 (2004), pp. 85–96. DOI: 10.1145/997150.997159.

[192] M. Sridharan, K. Tan, D. Bansal, and D. Thaler. “Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks, draft-sridharan-tcpm-ctcp-02”. 2008.

[193] Sudheer Poojary and Vinod Sharma. “Analysis of multiple flows using different high speed TCP protocols on a general network”. In: *Performance Evaluation* 104 (2016), pp. 42–62. ISSN: 01665316. DOI: 10.1016/j.peva.2016.08.002. arXiv: 1602.06653. URL: http://dx.doi.org/10.1016/j.peva.2016.08.002.

[194] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. “BBR: Congestion-Based Congestion Control”. In: *ACM Queue* 14.5 (2016), pp. 20–53. ISSN: 1542-7730. URL: http://doi.acm.org/10.1145/3012426.3022184.

[195] W3Techs. *Usage statistics and market share of Unix for websites*. W3Techs. Mar. 2018. URL: https://w3techs.com/technologies/details/os-unix/all/all.

[196] *Linux: The World’s Most Important Open Source Software Project*. The Linux Foundation. Mar. 2018. URL: https://www.linuxfoundation.org/about/.

[197] Pasi Sarolahti and Alexey Kuznetsov. “Congestion Control in Linux TCP”. In: *The FREENIX Track: 2002 USENIX Annual Technical Conference*. 2002, pp. 49–62. ISBN: 1880446014. DOI: 10.1029/2010GC003461.

[198] Yuchung Cheng. *Recent advancements in Linux TCP congestion control*. 2013. URL: https://datatracker.ietf.org/doc/slides-88-iccrg-6/.

[199] Mirja Kühlewind. *Effects of PRR after Slow Start*. Berlin, 2013. URL: https://www.ietf.org/proceedings/87/tcpm.html.

[200] Nasif Ekiz, Abuthahir Habeeb Rahman, and Paul D Amer. “Misbehaviors in TCP SACK generation”. In: *Computer Communication Review* 41.2 (2011), pp. 17–23. ISSN: 01464833 (ISSN). DOI: 10.1145/1971162.1971165. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84856463961%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=d4e7a82646cd82473a6f31eea0da3de2.

[201] Adam Dunkels. *Software: light-weight IP stacks*. Mar. 2018. URL: http://dunkels.com/adam/software.html.

[202] Contiki. *Contiki: The Open Source OS for the Internet of Things*. Contiki. Mar. 2018. URL: http://www.contiki-os.org/.

[203] EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. “mTCP: a Highly Scalable User-level TCP Stack for Multicore Systems”. In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA, USA, 2014, pp. 489–502. ISBN: 978-1-931971-09-6.

[204] V. Paxson, M Allman, J Chu, and M Sargent. *Computing TCP's Retransmission Timer, RFC 6298*. Tech. rep. Internet Engineering Task Force (IETF), June 2011, pp. 1–12. DOI: 10.17487/rfc6298. arXiv: arXiv:1011.1669v3. URL: https://www.rfc-editor.org/info/rfc6298.

[205] Yuchung Cheng and Neal Cardwell. *Packet loss detection based on recent acknowledgement (RACK)*. 2016. URL: http://www.tdcommons.org/dpubs%7B%5C_%7Dseries/192.

[206] N Cardwell and N Dukkipati. *RACK: a time-based fast loss detection algorithm for TCP draft-ietf-tcpm-rack-03*. 2018.

[207] M. Mathis. *Laminar TCP and the Case for Refactoring TCP Congestion Control*. 2012.

[208] Google. *TCP Laminar*. 2015. URL: https://developers.google.com/speed/protocols/tcp-laminar.

[209] *2018 Next-Gen Data Center Networking Report*. Tech. rep. SDxCentral, LLC, 2018, pp. 1–47. URL: https://www.sdxcentral.com/.

[210] *Corporate Hosting & Cloud Managed Services Trends*. Tech. rep. April. 451 Global Digital Infrastructure Alliance, 2017, pp. 1–8.

[211] IDC. *IDC FutureScape : Worldwide Cloud 2018 Predictions*. 2018.

[212] Mindy Cancila, Douglas Toombs, Alan D Waite, and Elias Khnaser. *2017 Planning Guide for Cloud Computing*. Tech. rep. October. Gartner, 2016, pp. 1–29.

[213] *2016 Trends in Datacenter Technologies*. Tech. rep. 451 Research, 2016, pp. 1–24.

[214] *Introducing data center fabric, the next-generation Facebook data center network*. Nov. 2014. URL: https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/.

[215] Jay Park. *Designing a Very Efficient Data Center*. 2015. URL: https://www.facebook.com/notes/facebook-engineering/designing-a-very-efficient-data-center/10150148003778920/.

[216] Yibo Zhu, Ben Y. Zhao, Haitao Zheng, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, and Ming Zhang. "Packet-Level Telemetry in Large Datacenter Networks". In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15* (2015), pp. 479–491. ISSN: 01464833. DOI: 10.1145/2785956.2787483. URL: http://dl.acm.org/citation.cfm?doid=2785956.2787483.

[217] Sushant Jain, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, Amin Vahdat, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, and Junlan Zhou. "B4: Experience with a Globally-Deployed Software Defined WAN". In: *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*. 2013, pp. 1–12. ISBN: 9781450320566. DOI: 10.1145/2486001.2486019. URL: http://dl.acm.org/citation.cfm?id=2486019%7B%5C%%7D5Cnhttp://dl.acm.org/citation.cfm?doid=2486001.2486019%7B%5C%%7D5Cnhttp://dl.acm.org/citation.cfm?doid=2486001.2486019.

[218] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang Hong Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. "Dynamo: Facebook's Data Center-Wide Power Management System". In: *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016* (2016), pp. 469–480. ISSN: 0163-5964. DOI: 10.1109/ISCA.2016.48.

[219] Dongyao Wu, Sherif Sakr, Liming Zhu, and Huijun Wu. "Towards Big Data Analytics across Multiple Clusters". In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2017), pp. 218–227. DOI: 10.1109/CCGRID.2017.73. URL: https://doi.org/10.1109/CCGRID.2017.73.

[220] *Global Industry 4.0 Readiness Report 2016*. Tech. rep. Danish Institute of Industry 4.0, 2016, pp. 1–157. URL: https://www.dii4.dk/.

[221] Data Center Knowledge. *Be Aware of These 5 Data Center Trends in 2018*. Informa USA, Inc. Jan. 2018. URL: http://www.datacenterknowledge.com/manage/be-aware-these-5-data-center-trends-2018.

[222] Data Center Journal. *Dynamic Global Data Center Market to Surge Through 2018*. Data Center Journal. Feb. 2018. URL: http://www.datacenterjournal.com/dynamic-global-data-center-market-surge-2018/.

[223] *Global warming: Data centres to consume three times as much energy in next decade, experts warn*. Independent. Jan. 2016. URL: https://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html.

[224] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, and William Lintner. *United States Data Center Energy Usage Report, LBNL-1005775*. Tech. rep. June. ERNEST ORLANDO LAWRENCE BERKELEY NATIONAL LABORATORY, 2016, pp. 1–66.

[225] Maria Avgerinou, Paolo Bertoldi, and Luca Castellazzi. "Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency". In: *Energies, MDPI* 10.10 (2017), pp. 1–18.

[226] *The green grid: get connected to efficient IT*. The Green Grid Association. 2018. URL: https://www.thegreengrid.org/.

[227] *Performance Indicator, Green Grid's New Data Center Metric, Explained*. Data Center Knowledge. July 2016. URL: http://www.datacenterknowledge.com/archives/2016/07/18/performance-indicator-green-grids-new-data-center-metric-explained.

[228] ITU-T. *JOINT COORDINATION ACTIVITY ON SOFTWARE DEFINED NETWORKS (JCA-SDN-D-001 Rev.6)*. Software-defined Networking (SDN). ITU-T. July 2017. URL: https://www.itu.int/en/ITU-T/sdn/Pages/default.aspx.

[229] *Telecommunications Infrastructure Standard for Data Centers (TIA-942)*. Telecommunications Industry Association (TIA). July 2017. URL: http://www.tiaonline.org/standards/buy-tia-standards.

[230] *ANSI/BICSI 002-2014, Data Center Design and Implementation Best Practices*. ANSI/BICSI. 2014. URL: https://www.bicsi.org/book_details.aspx?Book=BICSI-002-CM-14-v5&d=0.

[231] *Tiers Program Publications*. Uptime Institute. 2012 - 2018. URL: https://uptimeinstitute.com/resources.

[232] *INFORMATION TECHNOLOGY - GENERIC CABLING SYSTEMS - PART 5: DATA CENTRES*. European Committee for Electrotechnical Standardization (CENELEC). 2007. URL: https://standards.globalspec.com/std/1602396/cenelec-en-50173-5.

[233] *ISO/IEC 11801-5:2017 Information technology – Generic cabling for customer premises – Part 5: Data centres*. International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). 2017. URL: https://www.iso.org/isoiec-jtc-1.html,%20https://www.iso.org/standard/62247.html.

[234] L. Avramov and J. Rapp. *Data Center Benchmarking Terminology, RFC 8238*. Tech. rep. IETF, 2017, pp. 1–20.

[235] D. Stopp and B. Hickman. *Methodology for IP Multicast Benchmarking, RFC 3918*. Tech. rep. IETF, 2004, pp. 1–31.

[236] R. Mandeville and J. Perser. *Benchmarking Methodology for LAN Switching Devices, RFC 2889*. Tech. rep. IETF, 2000, pp. 1–35.

[237] *Software Defined Networking Research Group (sdnrg)*. Internet Research Task Force (IRTF), IETF. Nov. 2014. URL: https://datatracker.ietf.org/rg/sdnrg/about/.

[238] IRTF. *Network Function Virtualization Research Group (NFVRG)*. IETF, IRTF. 2018. URL: https://irtf.org/nfvrg.

[239] *P1916.1 - Standard for Software Defined Networking and Network Function Virtualization Performance*. PVE - Performance for Virtualized Environments. IEEE SDN. Dec. 2015. URL: http://standards.ieee.org/develop/project/1916.1.html.

[240] *Software Defined Virtual Networks project (NIST ISSMG)*. National Institue of Standards, Technology (NIST), Internet, and Scalable Systems Metrology Group. 2014. URL: https://www.nist.gov/programs-projects/software-defined-virtual-networks.

[241] *Network Function Virtualization*. ETSI, Industry Specification Group for Network Function Virtualization (ISG NFV). 2012. URL: http://www.etsi.org/technologies-clusters/technologies/nfv.

[242] *ONF Specifications*. Open Networking Foundation (ONF). 2009. URL: https://www.opennetworking.org/software-defined-standards/specifications/.

[243] Open Compute Project. *Collaborating to improve infrastructure design*. 2018. URL: http://www.opencompute.org/about/.

[244] *Open Source Networking: The LF Networking Fund (LFN)*. The Linux Foundation. Jan. 2018. URL: https://www.linuxfoundation.org/projects/networking/.

[245] *Open Networking Founddation: non-profit operator-led consortium*. Open Networking Foundation (ONF). 2018. URL: https://www.opennetworking.org/.

[246] *Introducing data center fabric, the next-generation Facebook data center network*. Nov. 2014. URL: https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/.

[247] *Introducing "Wedge" and "FBOSS," the next steps toward a disaggregated network*. Facebook. 2014. URL: https://code.facebook.com/posts/681382905244727/introducing-wedge-and-fboss-the-next-steps-toward-a-disaggregated-network/.

[248] Adel Abouchaev and Tim Laberge. *Brain-Slug : a BGP-Only SDN for Large-Scale Data-Centers*. Tech. rep.

[249] *SDN/NFV network standards don't address benefits of change*. TechTarget, SearchTelecom.com. Apr. 2016. URL: https://searchtelecom.techtarget.com/tip/SDN-NFV-network-standards-dont-address-benefits-of-change.

[250] B Constantine, G. Forget, R. Geib, and R. Schrage. *Framework for TCP Throughput Testing, RFC 6349*. Tech. rep. IETF, 2011, pp. 1–27.

[251] *Understanding Data Center Commissioning*. Tech. rep. Compass Datacenters, 2015, pp. 1–6.

[252] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities". In: *Proceedings of the 2009 International Conference on High Performance Computing and Simulation, HPCS 2009*. 2009, pp. 1–11. ISBN: 9781424449071. DOI: 10.1109/HPCSIM.2009.5192685. arXiv: 0907.4878.

[253] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya. "CloudAnalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications". In: *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*. 2010, pp. 446–452. ISBN: 9780769540184. DOI: 10.1109/AINA.2010.32.

[254] Saurabh Kumar Garg and Rajkumar Buyya. "NetworkCloudSim: Modelling parallel applications in cloud simulations". In: *Proceedings - 2011 4th IEEE International Conference on Utility and Cloud Computing, UCC 2011* Vm (2011), pp. 105–113. DOI: 10.1109/UCC.2011.24.

[255] Jongtack Jung and Hwangnam Kim. "MR-CloudSim: Designing and implementing MapReduce computing model on CloudSim". In: *International Conference on ICT Convergence*. 2012, pp. 504–509. ISBN: 9781467348287. DOI: 10.1109/ICTC.2012.6387186.

[256] Jungmin Son, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, Xiaohui Ji, Young Yoon, and Rajkumar Buyya. "CloudSimSDN: Modeling and simulation of software-defined cloud data centers". In: *Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015*. 2015, pp. 475–484. ISBN: 9781479980062. DOI: 10.1109/CCGrid.2015.87.

[257] Sareh Fotuhi Piraghaj, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya. "ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers". In: *Software: Practice and Experience* (2016). ISSN: 00380644. DOI: 10.1002/spe.2422. arXiv: 1008.1900. URL: http://doi.wiley.com/10.1002/spe.2422.

[258] Minxian Xu, Guozhong Li, Wutong Yang, and Wenhong Tian. "FlexCloud: A flexible and extendible simulator for performance evaluation of virtual machine allocation". In: *Proceedings - 2015 IEEE International Conference on Smart City, SmartCity 2015, Held Jointly with 8th IEEE International Conference on Social Computing and Networking, SocialCom 2015, 5th IEEE International Conference on Sustainable Computing and Communic* (2015), pp. 649–655. DOI: 10.1109/SmartCity.2015.143. arXiv: 1501.05789.

[259] A Malik, K Bilal, S Malik, Z Anwar, K Aziz, D Kliazovich, N Ghani, S Khan, and R Buyya. "CloudNetSim++: A GUI Based Framework for Modeling and Simulation of Data Centers in OMNeT++". In: *IEEE Transactions on Services Computing* PP.99 (2015), p. 1. ISSN: 1939-1374. DOI: 10.1109/TSC.2015.2496164.

[260] Chengchen Hu, Ji Yang, and Zhiming Gong. “DesktopDC : Setting All Programmable Data Center Networking Testbed on Desk”. In: *Proceedings of the 2014 ACM conference on SIGCOMM*. Chicago, Illinois, USA, 2014, pp. 593–594.

[261] Zhangxi Tan, Krste Asanovic, and David Patterson. “Datacenter-Scale Network Research on FPGAs”. In: *Exascale Evaluation and Research Techniques Workshop (EXERT)*. Newport Beach, CA, USA, 2011, pp. 1–6.

[262] *OpenStack: Open source software for creating private and public clouds*. The OpenStack project. 2018. URL: https://www.openstack.org/.

[263] Mary Branscombe. *Microsoft Runs a Simulation of the Entire Azure Network to Prevent Outages*. Data Center Knowledge. Nov. 2017. URL: http://www.datacenterknowledge.com/networks/microsoft-runs-simulation-entire-azure-network-prevent-outages?NL=DCK-01&Issue=DCK-01_20171108_DCK-01_206&sfvc4enews=42&cl=article_1_b&utm_rid=CPNET000000117705&utm_campaign=5614&utm_medium=email&elq2=f5041ca0594f4126aac25f2e96e2e6a5.

[264] *Data center test bed facility opens*. DatacenterDynamics. Oct. 2007. URL: http://www.datacenterdynamics.com/data-center-test-bed-facility-opens/30242.fullarticle.

[265] *SICS ICE – data center in Luleå*. RISE SICS North. Feb. 2016. URL: https://www.sics.se/projects/sics-ice-data-center-in-lulea.

[266] Bailey Laura Navrátil Milan and Charlie Boyle. *Red Hat Enterprise Linux 7 Performance Tuning Guide*. Tech. rep. Red Hat Inc., 2018, pp. 1–115.

[267] Jeremy Eder. *Low Latency Performance Tuning Guide for Red Hat Enterprise Linux 7*. Tech. rep. February. Red Hat Inc., 2015.

[268] Fotios K Liotopoulos. “Energy-Efficient Simulation and Performance Evaluation of Large-Scale Data Centers”. In: (2015), pp. 3121–3127.

[269] *ONOS - Open Network Opearing System*. Open Networking Foundation. 2018. URL: https://onosproject.org/.

[270] *OpenDaylight: Open Source SDN Platform*. 2017. URL: https://www.opendaylight.org/.

[271] Ryu SDN Framework Community. *Ryu: component-based software defined networking framework*. 2018. URL: https://osrg.github.io/ryu/ (visited on 01/03/2018).

[272] *OF-CONFIG server for Open vSwitch*. Open Networking Foundation. 2015-2018. URL: https://github.com/openvswitch/of-config,%20https://www.opennetworking.org/software-defined-standards/models-apis/.

[273] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. *Network Configuration Protocol (NETCONF), RFC 6241*. Tech. rep. IETF, 2011, pp. 1–113. arXiv: arXiv:1011.1669v3.

[274] *rConfig: network management, configuration management*. rConfig. 2012-2017. URL: https://www.rconfig.com/.

[275] *Achieve speed, scale, and consistency by automating your infrastructure with Chef*. Chef Software, Inc. 2018. URL: https://www.chef.io/chef/.

[276] *The Puppet approach*. Puppet. 2018. URL: https://puppet.com/products/how-puppet-works.

[277] *ANSIBLE OPEN SOURCE*. Red Hat Inc. 2018. URL: https://www.ansible.com/.

[278] *Nagios open source: the industry standard in IT infrastructure monitoring*. Nagios Enterprises, LLC. 2009-2018. URL: https://www.nagios.org/.

[279] *OpenNMS: the network monitoring platform*. The OpenNMS Group, Inc. 2017. URL: https://opennms.org/en/opennms.

[280] *Zabbix: Monitor anything. Solutions for any kind of IT infrastructure, services, applications, resources*. Zabbix LLC. 2001-2018. URL: https://www.zabbix.com/.

[281] *Cacti: the complete RRDtool-based graphing solution*. The Cacti Group, Inc. 2004-2018. URL: https://www.cacti.net/.

[282] *How to Use Grafana with InfluxDB to Monitor Time Series Data*. InfluxData Inc. 2018. URL: https://www.influxdata.com/blog/how-to-use-grafana-with-influxdb-to-monitor-time-series-data/.

[283] *statEngine: real-time Open Source Data Analytics and Visualization Platform*. National Institute of Standards and Technology. Jan. 2018. URL: https://www.nist.gov/ctl/pscr/real-time-open-source-data-analytics-and-visualization-platform.

[284] M. Chandramouli, B. Claise, B. Schoening, J. Quittek, and T. Dietz. *Monitoring and Control MIB for Power and Energy, RFC 7460*. Tech. rep. IETF, 2015, pp. 1–69.

[285] *Power consumption via SNMP*. Cisco. July 2010. URL: https://supportforums.cisco.com/t5/security-management/power-consumption-via-snmp/td-p/1444263.

[286] *Juju: operate big software at scale on any cloud*. Canonical Group Ltd. 2018. URL: https://jujucharms.com/.

[287] *Docker*. Docker Inc. 2018. URL: https://www.docker.com/.

[288] *Kubernetes: Production-Grade Container Orchestration*. The Kubernetes Authors. 2018. URL: https://kubernetes.io/.

[289] *HPCC Systems (High Performance Computing Cluster): end-to-end big data in a massively scalable supercomputing platform*. HPCC Systems. 2011-2018. URL: https://github.com/hpcc-systems,%20https://hpccsystems.com/.

[290] *Apache Hadoop*. The Apache Software Foundation. 2008. URL: http://hadoop.apache.org/.

[291] *Apache Spark: Lightning-fast unified analytics engine*. The Apache Software Foundation. 2009. URL: http://spark.apache.org/.

[292] *Apache Storm*. The Apache Software Foundation. 2015. URL: http://storm.apache.org/.

[293] *MapD: the extreme analytics platform*. MapD Technologies, Inc. 2018. URL: https://www.mapd.com/.

[294] *InfluxDB: time series database*. InfluxData Inc. 2018. URL: https://www.influxdata.com/time-series-platform/influxdb/.

[295] Andrew Collette et al. *HDF5 for python*. 2014-2018. URL: https://www.h5py.org/.

[296] *MongoDB for giant ideas*. NoSQL based database. MongoDB, Inc. 2018. URL: https://www.mongodb.com/.

[297] *Apache HBase: the Hadoop database, a distributed, scalable, big data store*. The Apache Software Foundation. 2007. URL: https://hbase.apache.org/.

[298] *Graphite*. Orbitz. 2008. URL: https://graphiteapp.org/.

[299] *Telegraf: Agent for Collecting and Reporting Metrics and Data*. InfluxData, Inc. 2018. URL: https://www.influxdata.com/time-series-platform/telegraf/.

[300] P. Phaal, S. Panchen, and N. McKee. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, RFC 3176 (Informational)*. Tech. rep. IETF, 2001, pp. 1–31.

[301] *SFlow-RT: real-time analytics for actionable intelligence to support DevOps, Orchestration and SDN applications*. InMon Corp. 2015-2018. URL: https://sflow-rt.com/.

[302] *Host sFlow: Data Center Wide Server Performance Monitoring*. Host sFlow. 2010-2018. URL: https://sflow.net/about.php.

[303] *TensorFlow: An open source machine learning framework for everyone*. Google, community. 2017-2018. URL: https://www.tensorflow.org/.

[304] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://scikit-learn.org/stable/.

[305] *The Microsoft Cognitive Toolkit (CNTK)*. Microsoft. 2018. URL: https://www.microsoft.com/en-us/cognitive-toolkit/.

[306] *Apache MXNet (Incubating): A flexible and efficient library for deep learning*. The Apache Software Foundation (ASF). 2018. URL: https://mxnet.incubator.apache.org/.